VISOR

(Variable Interval Schedule Of Reinforcement)

System Documentation

prepared for

Dr. Daniel W. Scott III

by

Daniel Paul Long

December 12, 1979

My special thanks to Dr. Daniel Scott, my Computer Sciences faculty advisor; Dr. Ernest Harrell, my Psychology faculty advisor; John Giles and Fransisco Bascunan, Department of Computer Sciences employees and friends who helped me with the circuit boards.

Paul Long

Contents

Documentation

on

The VI Schedule of Reinforcement Program

prepared for

Dr. Daniel W. Scott III
CSCI 488

by

Daniel Paul Long
Oct. 6, 1979

Functional Description

of

The VI Schedule of Reinforcement Program

prepared for

Dr. Daniel W. Scott III

CSCI 488

by

Daniel Paul Long

Oct. 4, 1979

Functional Description

of

The VI Schedule of Reinforcement Program

This program will be used in operant behavior research to monitor and record responses and trigger and record reinforcements on a variable reinforcement (VI) schedule.

The original application of this program will be the servicing of several rat cages simultaneously. The response will be the pressing of a metal bar in the cage, the reinforcement will be the triggering of a feeding mechanism which disperses a food pellet into the cage. The subsequent applications of this program are not limited, in that the actual response and reinforcement devices and the subject type are all treated indifferently by the program.

During a session, all responses and reinforcements are recorded. At the start of a session, a timer is initialized with a time interval varying about a user given mean and standard deviation under a Gaussian distribution. When the time interval has elapsed, the next response receives a reinforcement. At the same time, the timer is reinitialized with another time interval varying in the same manner as before. This procedure is repeated until the end of the session. During the session, if a given number of reinforcements are reached, three things occur. They are: 1) reinforcemants

are stopped, 2) responses are no longer counted, and 3) a lamp is lit indicating which unit reached its maximum number of reinforcements.

The session can be terminated before the specified time interval has elapsed by pressing the reset button on the computer, whereupon the program can be restarted. The program comes back up with the output table that would have come up if the session had terminated normally. The data accumulated is from the point the reset button was pressed.

These procedures are performed for from one to eight separate units independently and concurrently. Each unit has its number of responses and number of reinforcements stored separately. All units are serviced during a common session--they cannot be serviced over different sessions, starting and stopping at different times, with respect to each other. Each unit has a user-supplied mean time interval, standard deviation, and maximum number of reinforcements associated with it.

The user of this program is to input the following information before each session:

Item	Range	Units	Precision
-Session length (optional, default is 30 minutes)	1 to 255	minutes	1 minute

(table continued next page)

(table continued)

Item	Range	Units	Precision
-Total number of units to be serviced during session. (optional, default is 4 units)	1 to 8	na	na
-Unit number with which the following items are to be associated.	1 to above item		na
-Mean time interval	0 to 255	seconds	1 second
-Standard deviation	0 to 85	seconds	1 second
-Maximum number of reinforcements allowed.	0 to 999	na	na

A unit to be used for continuous reinforcement (CRF) need only to be assigned a mean time interval of zero. A very large (in fact, 65,535) maximum number of reinforcements allowed is obtained by assigning zero to that unit.

If the total-number-of-units item is to be specified, (default is 4 units), it must be entered before any other item. Otherwise, the items may come in any order. A particular item may be changed by entering it again.

A session is started by entering an 'S', but only after all items have been entered. At the end of each

session, the following is printed out for each unit:

Item	Range
Number of responses	0 to 65,535
Number of reinforcements	0 to 65,535

After this is printed, a message is printed, asking if another session is to be run. If the user's reply is affirmative, the inputting procedure is repeated. If the user's reply is negative, control is transfered to the monitor program.

The formats for inputting and outputting information on the terminal are shown on an accompanying page. If invalid characters or out of range values are entered at the terminal, no explicit error message is printed indicating the type of error committed. For all errors, a question mark is printed, whereupon the user should reenter the proper character(s), referring to the user documentation first, if neccessary.

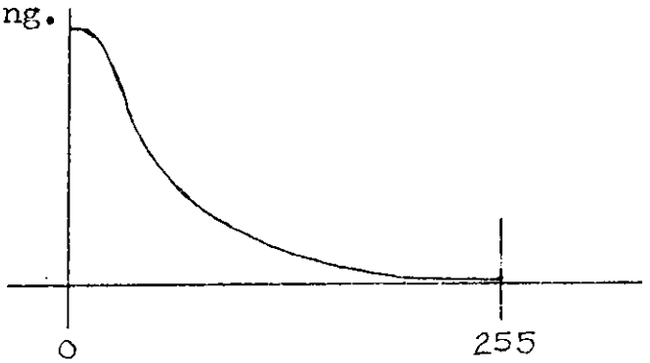
Data entered or generated for a particular session is destroyed upon execution of another inputting phase after the session is over.

The method for generating varying time intervals involves retrieving random normally distributed values from a large static table[†] and adjusting them to the

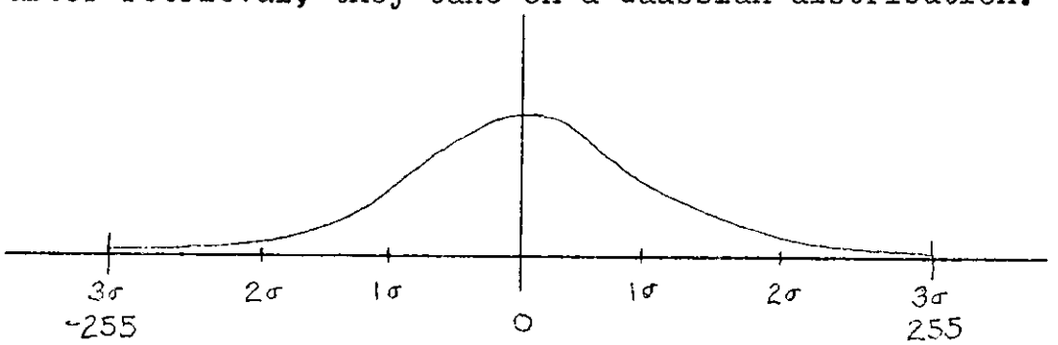
[†] This table is created by the PL/I program NORMHEX.

particular unit's dispersion.

This table contains unsigned integers from 0 to 255. The abscissa of the following graph is the value, while the ordinate is the relative probability of the value occuring.



The values do not comprise a Gaussian distribution as they are. By assigning signs to the values randomly, after retrieval, they take on a Gaussian distribution.



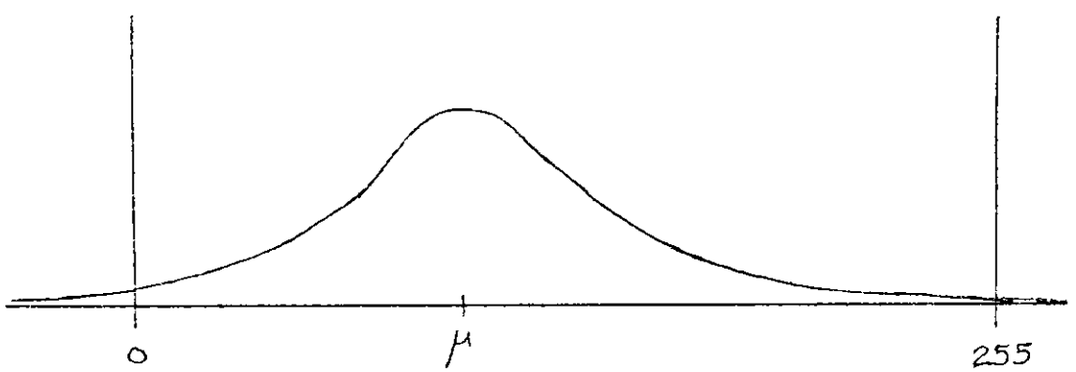
The table is set up so that, if we were talking about the eventual signed aspect that the values take on, 255 (and likewise -255) occurs at three standard deviations out from the mean of zero.

The user specifies the dispersion of the time intervals for each unit before the session, by defining one standard deviation out from the unit's mean of, for example, 50 seconds, to be so many seconds of, for example,

20 seconds.

The program retrieves a sample from the table, say 68, and multiplies it by the 20 seconds. This product, in this case 1360, is multiplied by three to give 4080, then divided by 256 to give 15 (the fraction is always truncated.) The divisor should be 255, but since 256 is easier and faster to implement and execute and the introduced error is small,† speed is chosen over accuracy. A sign is randomly assigned to the result, viz 15, so that, for example, 15 is subtracted from the unit's mean of 50 seconds, giving a time interval of 35 seconds. Alternately, 15 could be added to 50 seconds giving 65 seconds. This final result is then placed in the unit's timer.

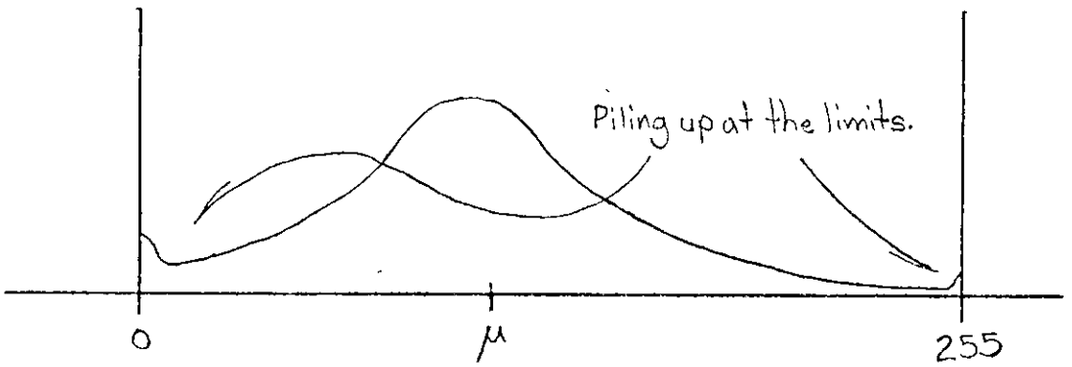
If a final result is less than zero, zero is placed in the unit's timer instead. If a final result is greater than 255, 255 is placed in the unit's timer instead. This effects the distribution of time intervals in the following way:



The above distribution becomes the following:

(continued on next page)

†ie. .39 percent error.



This distortion or "piling up" near zero and 255 increases as the user defined value of one standard deviation increasesand as the user defined mean becomes very small or very large. As long as these two parameters remain within reasonable, or rather practical, limits, this distortion should be insignificant.

More formally, the entire method is as follows:

Where S = sample from the table
and where, for each unit, T = time interval
SD = user defined value for 1 standard deviation
VI = user defined mean

Calculate T = S * SD * 3 / 256
Truncate T
Randomly alternate:
a) Set T = VI + T
b) Set T = VI - T
If T < 0, set T = 0
If T > 255, set T = 255
Then T has the time interval on the interval (0,255)

Starting from the bottom of the table, samples are retrieved sequentially. After all samples have been used, retrieval continues from the bottom of the table again.

The samples are distributed in the table randomly, but since all unit's retrieve samples from the same table and each unit performs its retrieval virtually randomly with respect to the other units, the retrieval process increases the table's inherent randomness.

This program is written in M6800 assembly language. The hardware required for the operation of this program include:

1) M6800-based computer system, such as the SWTPC 6800 computer system.
2) Computer terminal, such as the Decwriter hard copy terminal or the Lear-Siegler ADM-3A CRT terminal.
3) Electrical response and reinforcement devices, such as those in a rat chamber.
4) Interface between the computer and the response/ reinforcement devices; custom built, most likely.†
5) Electrical device supplying an interrupt signal to the computer.††
6) Indicators interfaced with the computer, such as LEDs (light emitting diodes.)††

† See the paper: The Parallel Input/Output Board, by P. Long
†† See the paper: The Presettable Interrupt Frequency Generator/ Audio-Visual Indicator Board, by P. Long

An accompanying diagram illustrates the relationship these components have to each other.

The device described in number 5, above, is the source of the clock signal used by the program to measure time. The clock signal is a 20 hz. time base.

The final version of the program may be placed in ROM. (ROM stands for read-only-memory and is different from most other memory devices in that it retains its contents even after power is removed from it. It also cannot be written over.) The program will, therefore, not have to be loaded into the computer prior to using it.

The program will be tested by first manipulating the response device (metal bar) in one rat cage by hand while the reinforcement device (feeder) is under control of the program. When the program monitors and controls this one rat cage as prescribed, the remaining cages will be serviced with rats in all cages.

DIALOGUE SUMMARY

underlined characters mean you are to type them.

^ means just 1 space.

△ means any number of spaces (0 to ∞).

(Note: Begin execution of VISORP at location 0600_X.)

VISORP

{
△#△number-of-units^
△unit-number^△mean^△deviation^△reinforcement-maximum^
△T△session-length^
{△START / △S?}
△X †
△^ ††
}

(Note: To terminate VISORP early, press reset button then restart execution of VISORP at location 0947_X.)

{FINISH / EARLY FINISH}

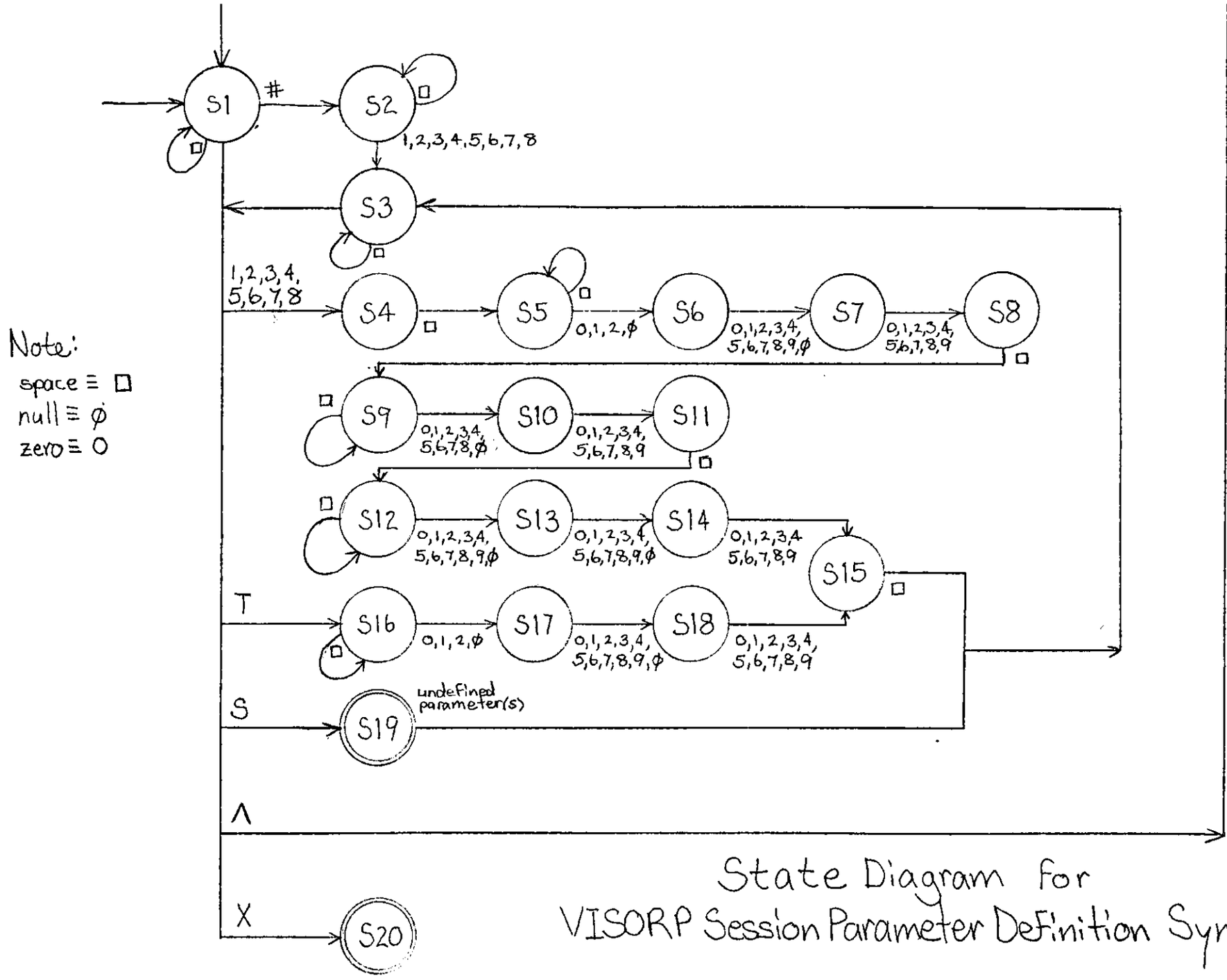
CHAMBER	RESPONSES	REINFORCEMENTS
unit-number	unit's responses	unit's reinforcements
.	.	.
.	.	.
.	.	.

ANOTHER?△reply †††

† X causes a jump to the computer systems monitor.

†† ^, or circumflex, restarts the entire inputting phase; all previously entered parameters are neglected.

††† Reply is either Y for yes or N for no.



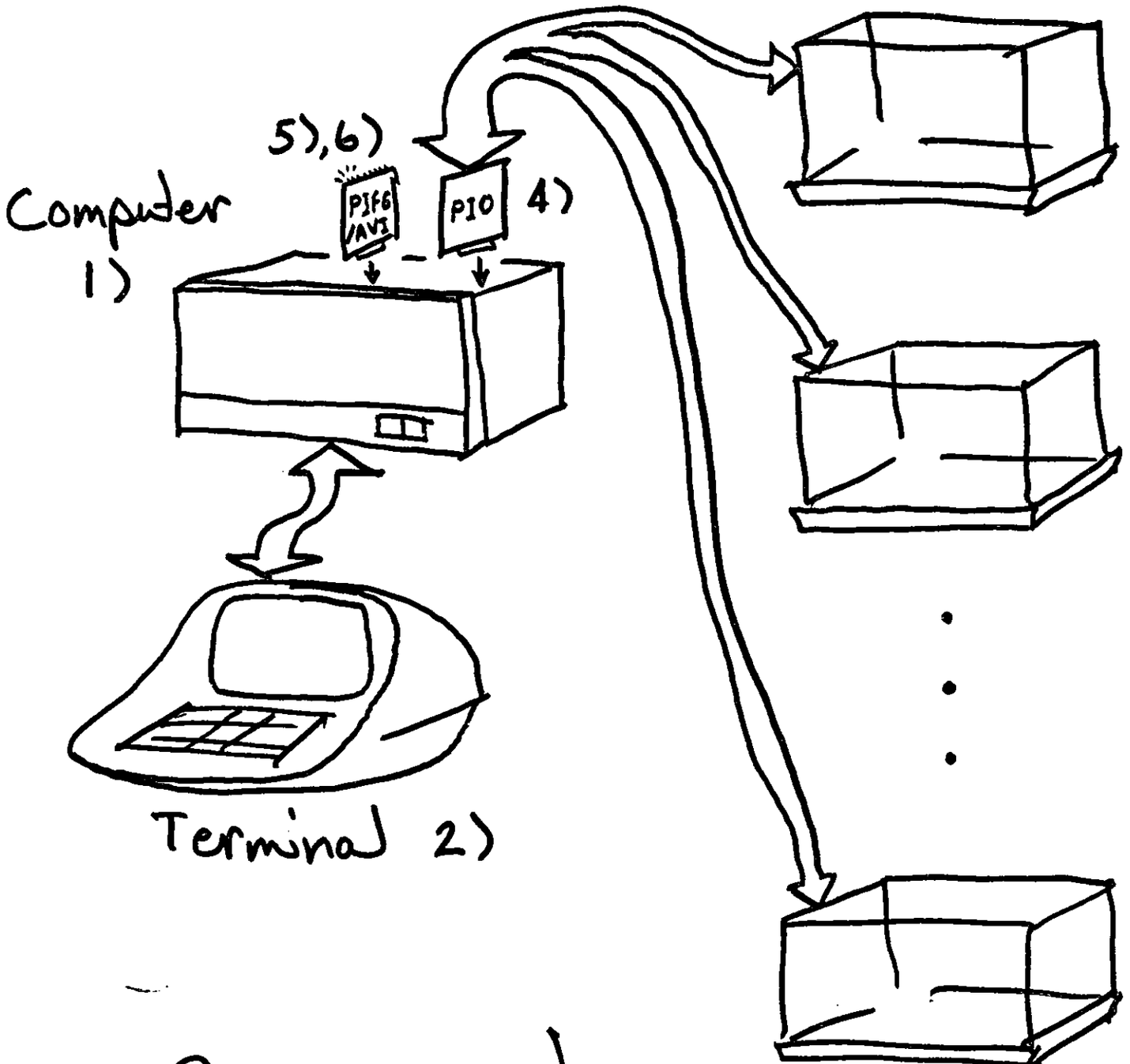
Note:
space ≡ □
null ≡ ϕ
zero ≡ O

State Diagram for
VISORP Session Parameter Definition Syntax

```
VISORP

#5
2 100 49 500
1 60 30 400
3 50 25 999
4 R?
4 255 85 0
4 255 85 999
T  120
6?
5 100 75 400
4 230?
5?
4 126 84 399
2 0 0 0
START

FINISH
```

CHAMBER	RESPONSES	REINFORCEMENTS
1	3020	1250
2	2000	433
3	200	32
4	4056	200
5	521	319

```
ANOTHER? N
*
```



Component Diagram

VISORP

program listing and output

prepared for

Dr. Daniel W. Scott III
CSCI 488

by

Daniel Paul Long
May 23, 1979

```
00001    0000                    ORG   0
00002                    ********************************************************************************
00003                    *                                TABLES                                        *
00004                    ********************************************************************************

00006    0000   0008     VI      RMB   8          VI IN 1-BYTE / UNIT
00007    0008   0008     SD      RMB   8          STD. DEV IN 1-BYTE / UNIT
00008    0010   0010     REINMX  RMB   16         MAX. # OF REINFORCEMENTS ALLOWED; 2-BYTE / UNIT
00009    0020   0010     RESPS   RMB   16         # OF RESPONSES IN 2-BYTE / UNIT
00010    003C   0010     REINFS  RMB   16         # OF REINFORCEMENTS IN 2-BYTE / UNIT
00011    0040   0008     CLOCK   RMB   8          TIMING DATA; 1-BYTE / UNIT
00012    0048   0008     FREQDV  RMB   8          DIVIDE INT. HZ. DOWN TO 1 HZ.; 1-BYTE / UNIT


00014                    ********************************************************************************
00015                    *                                FLAGS                                         *
00016                    ********************************************************************************

00018    005C   0001     RMXFLG  RMB   1          RAISED WHEN REINMX = REINFS; 1 BIT  / UNIT
00019    0051   0001     LEDFLG  RMB   1          RAISED WHEN REINMX = REINFS; 1 BIT / UNIT
00020    0052   0001     RENFLG  RMB   1          RAISED WHEN A UNIT NEEDS REINFORCING; 1 BIT  / UNIT
00021    0053   0001     ALLTST  RMB   1          RAISED WHEN INPUT LINE HAS BEEN COMPLETED; 1 BIT /
00022    0054   0001     TRURSP  RMB   1          TRUE RESPONSE DATA FROM INTRPT TO RSPCHK SR'S
00023    0055   0001     WAIFLG  RMB   1          USED IN A WAIT-FOR-INTERRUPT LOOP


00025                    ********************************************************************************
00026                    *                              VARIABLES                                       *
00027                    ********************************************************************************

00029    005E   0001     SFRQDV  RMB   1          DIVIDE INTERRUPT HZ. DOWN TO 1 HZ. FOR SESSION TIME
00030    0057   0001     STIME   RMB   1          SESSION TIME; LENGTH OF SESSION IN MINUTES
00031    005E   0001     MINUTE  RMB   1          USED TO DIVIDE THE 1 HZ. SIGNAL DOWN TO 1 MIN./SEC.
00032    0059   0001     PRVRSP  RMB   1          RESPONSES' RECENT HISTORY
00033    005A   0001     BITPOS  RMB   1          TEMPORARY STORAGE IN RENCHK SR
00034    005B   0002     COUNT   RMB   2          USED IN OUTPUT SR
00035    005D   0002     OFFSET  RMB   2          GENERAL VARIABLE
00036    005F   0002     SAVEX   RMB   2          SAVE X REGISTER
00037    0061   0002     NUMBER  RMB   2          PRINT SR USES TO SAVE THE BINARY DATA TO BE PRINTED
00038    0063   0003     NUM     RMB   3          PRINT SR USES TO STORE RESULT OF CONV. TO BCD
00039    0066   0001             RMB   1          X REG. GETS UNITS FROM HERE (ALWAYS 0)
00040    0067   0001     UNITS   RMB   1          NUMBER OF UNITS TO BE SERVICED DURING SESSION
00041    0068   0001     UNITSP  RMB   1          CONTAINS BIT POSITION OF # OF UNITS
00042    006A   0002     POINTR  RMB   2          INITIAL OFFSET TO BE USED FOR INTERV TABLE
00043    006B   0001     TRIGS   RMB   1          CONTAINS UNITS' BIT(S) USED TO TRIGGER REINFORCEMNT
00044    006C   0002             RMB   2          PART OF INTERM
00045    006E   0003     INTERM  RMB   3          NOLDGO SR USES TO RECIEVE INPUT W/OUT LEADING ZEROS
00046    0071   0001     MCND    RMB   1          MULTIPLICAND STORAGE FOR DEVIAT SR
00047    0072   0001     MPLR    RMB   1          MULTIPLIER STORAGE FOR DEVIAT SR
00048    0073   0001     MPYCNT  RMB   1          COUNTS # OF SHIFTS IN DEVIAT SR
00049    0074   0001     TEMPA   RMB   1          TEMPORARY STORAGE FOR A IN DEVIAT & TPARAM SRS
```

```
00050  0075  0001        TEMPB  RMB  1          TEMPORARY STORAGE FOR B IN DEVIAT SR
00051  0076  0004        RNDSGN RMB  4          HOLDS BITS FOR RANDOM SIGN ASSIGNMENT IN DEVIAT SR
00052  007A  0001        SAVVI  RMB  1          TEMPORARY STORAGE OF UNIT'S VI DURING DEVIAT SR
00053  007B  0001        PRVMAX RMB  1          LEDFLG'S RECENT HISTORY
00054  C07C  0001        RSPFLG RMB  1          TEMP STORAGE OF TRURSP BETWEEN RSPCHK & RENCHK SR'S

00056  C0A0               ORG  $A0

00058        0400  K      EQU  1024
00059  00A0  0400  SAMPLE RMB  1*K            MEMORY RESERVED FOR STATIC NUMERICAL RESOURCE USED
00060              *                            BY DEVIAT SR TO PRODUCE NORM. DISTR. VALUES

00062              * PRINT SR USES TO CONVERT BINARY TO DECIMAL VIA TABLE LOOK-UP

00064  04A0  00    TABLE  FCB  $0,$0,$1,$0,$0,$2,$0,$0,$4
       04A1  00
       04A2  01
       04A3  00
       04A4  00
       04A5  02
       04A6  00
       04A7  00
       04A8  04
00065  04A9  00         FCB  $0,$0,$8,$0,$0,$16,$0,$0,$32
       04AA  00
       04AB  08
       04AC  00
       04AD  00
       04AE  16
       04AF  00
       04B0  00
       C4B1  32
00066  04B2  00         FCB  $0,$0,$64,$0,$1,$28,$0,$2,$56
       04B3  00
       04B4  64
       04B5  00
       04B6  01
       04B7  28
       04B8  00
       04B9  02
       04BA  56
00067  04BB  00         FCB  $0,$5,$12,$0,$10,$24,$0,$20,$48
       04BC  05
       04BD  12
       04BE  00
       04BF  10
       04C0  24
       04C1  00
       04C2  20
       04C3  48
00068  04C4  00         FCB  $0,$40,$96,$0,$81,$92,$1,$63,$84
       04C5  40
```

```
                04C6  96
                04C7  00
                04C8  81
                04C9  92
                04CA  01
                04CB  63
                04CC  84
00069           04CD  03            FCB   $3,$27,$68
                04CE  27
                04CF  68


00071           ********************************************************************************
00072           *                                MESSAGES                                      *
00073           ********************************************************************************

00075           04D0  0D    MSG1    FCB   $D,$A,$A,$A,$A,$A
                04D1  0A
                04D2  0A
                04D3  0A
                04D4  0A
                04D5  0A
00076           04D6  56            FCC   6,VISORP
                04D7  49
                04D8  53
                04D9  4F
                04DA  52
                04DB  50
00077           04DC  0D            FCB   $D,$A,$A,4
                04DD  0A
                04DE  0A
                04DF  04
00078           04E0  54    MSG2    FCC   4,TART
                04E1  41
                04E2  52
                04E3  54
00079           04E4  0D            FCB   $D,$A,$A,4
                04E5  0A
                04E6  0A
                04E7  04
00080           04E8  46    MSG3    FCC   6,FINISH
                04E9  49
                04EA  4E
                04EB  49
                04EC  53
                04ED  48
00081           04EE  0D            FCB   $D,$A,$7,4
                04EF  0A
                04F0  07
                04F1  04
00082           04F2  43    MSG4    FCC   7,CHAMBER
                04F3  48
                04F4  41
                04F5  4D
```

```
                04F6    42
                04F7    45
                04F8    52
00083           04F9    04              FCB     4
00084           04FA    52      MSG5    FCC     9,RESPONSES
                04FB    45
                04FC    53
                04FD    50
                04FE    4F
                04FF    4E
                0500    53
                0501    45
                0502    53
00085           0503    04              FCB     4
00086           0504    52      MSG6    FCC     14,REINFORCEMENTS
                0505    45
                0506    49
                0507    4E
                0508    46
                0509    4F
                050A    52
                050B    43
                050C    45
                050D    40
                050E    45
                050F    4E
                0510    54
                0511    53
00087           0512    0D              FCB     $D,$A,4
                0513    0A
                0514    04
00088           0515    0D      MSG7    FCB     $D,$A
                0516    0A
00089           0517    41              FCC     8,ANOTHER?
                0518    4E
                0519    4F
                051A    54
                051B    48
                051C    45
                051D    52
                051E    3F
00090           051F    04              FCB     4
00091           0520    0A      MSG8    FCB     $A,$D
                0521    0D
00092           0522    45              FCC     6,EARLY
                0523    41
                0524    52
                0525    4C
                0526    59
                0527    20
00093           0528    04              FCB     4
00094           0529    3F      QUEST   FCC     1,?
00095           052A    0D      CRLF    FCB     $D,$A,$4
                052B    0A
                052C    04
```

```
00097                          * LOCATIONS IN MONITOR

00099  A000    IOV    EQU  $A000
00100  E07E    PDATA1 EQU  $E07E
00101  E0CC    OUTS   EQU  $E0CC
00102  E0E3    MONITR EQU  $E0E3
00103  E1AC    INEEE  EQU  $E1AC
00104  E1D1    OUTEEE EQU  $E1D1


00106                          * I/O PORT ADDRESSES

00108  8012    MAXOUT EQU  $8012      LEDS & BEEPER OUTPUT
00109  8018    INRSPS EQU  $8018      RAT CAGE INTERFACE INPUT
00110  801A    OUTRNF EQU  INRSPS+2   RAT CAGE OUTPUT


00112  0014    DIVISR EQU  20         DIVIDE 20 HZ. DOWN TO 1 HZ.
00113  001E    STDFLT EQU  30         SESSION TIME DEFAULT: 30 MINUTES
00114  0004    UDFLT  EQU  4          # OF UNITS DEFAULT: 4 UNITS
00115  0400    TBLSIZ EQU  1*K        NUMBER OF BYTES IN SAMPLE TABLE
00116  520D    FIRST2 EQU  $520D      INITIAL VALUE TO BE STORED IN RNDSGN
00117  CBB9    SECND2 EQU  $CBB9
```

```
00119                           NAM   VISORP VERSION 05.23.79
00120                           OPT   MEMORY,SYMBOL

00122   0600                    ORG   $0600



00124                   ********************************************************************************
00125                   *                              VISORP MAIN PROGRAM                             *
00126                   ********************************************************************************

00128                   *    THIS CODE SEGMENT IS THE HIGHEST LEVEL OF CONTROL IN VISORP--THE MAJOR
00129                   *  COMPONENTS ARE CALLED FROM HERE. VISORP (VARIABLE INTERVAL SCHEDULE OF
00130                   *  REINFORCEMENT PROGRAM) IS THE SOFTWARE IN A REAL-TIME COMPUTING SYSTEM USED
00131                   *  IN OPERANT BEHAVIOR RESEARCH TO EXECUTE A VI (VARIABLE INTERVAL) SCHEDULE
00132                   *  FOR UP TO EIGHT SUBJECTS, SIMULTANEOUSLY. IN THIS REGARD, THE SUBJECTS ARE IN
00133                   *  A TIME-SHARING ENVIRONMENT DUE TO THE HIGH SPEED OF THE COMPUTER AS IT POLLS
00134                   *  A SYSTEM OF FLAGS IN THE DSPTCH (DISPATCH) PART OF VISORP AND ACTS
00135                   *  ACCORDINGLY. BEFORE EACH RUN, THE USER DEFINES DIFFERENT ASPECTS OF THE
00136                   *  SESSION IN THE INPUT SR. WHEN THE SESSION IS OVER, THE TERMINAL PRINTS THE
00137                   *  RESULTS IN THE FORM OF HOW MANY TIMES EACH SUBJECT RESPONDED AND HOW MANY
00138                   *  TIMES EACH SUBJECT RECEIVED A REINFORCEMENT. (IN THE ORIGINAL APPLICATION,
00139                   *  RESPONSES ARE THE PRESSING OF A METAL BAR BY A RAT AND REINFORCEMENTS ARE
00140                   *  THE FEEDING OF FOOD PELLETS TO THE RAT.) AFTER THE DATA IS PRINTED, ANOTHER
00141                   *  SESSION MAY BE RUN.

00143   0600    01      VISORP NOP
00144   0601    0F             SEI                     |       INTERRUPTS MASKED
00145   0602    8D  1D  SESHUN BSR   INITAL            | CONFIGURE INTERFACES
00146   0604    8D  35         BSR   CLEAR             | CLEAR & INITIALIZE VARIABLES
00147   0606    8D  64         BSR   INPUT             | RECEIVE VI PARAMETERS FROM TERMINAL
00148   0608    0E             CLI                     |      INTERRUPTS UNMASKED

00150   0609    96  55  DSPTCH LDA A WAIFLG            WAIT FOR INTERRUPT SR TO EXECUTE
00151   060B    27  FC         BEQ   DSPTCH
00152   060D    7F  0055       CLR   WAIFLG

00154   0610    BD  0858       JSR   IO                PROCESS RESPONSES AND REINFORCEMENTS
00155   0613    BD  088A       JSR   RSPCHK            CHECK TRUE RESPONSES
00156   0616    BD  088F       JSR   RENCHK            CHECK IF ANY REINFORCEMENTS NEED TRIGGERING
00157   0619    BD  08F1       JSR   MAXCHK           CHECK IF ANY UNIT LED'S NEED LIGHTING
00158   061C    BD  0900       JSR   BIGBEN           MAINTAIN CLOCKS
00159   061F    20  E8         BRA   DSPTCH           LOOP BACK

00161                   ************************ END OF VISORP ***********************************
```

```
00163                    ********************************************************************************
00164                    *                                INITAL SR                                     *
00165                    ********************************************************************************

00167                    *    THIS SR IS CALLED BY VISORP MAIN PROGRAM TO INITIALIZE (CONFIGURE) THE TWO
00168                    *  INTERFACE BOARDS USED BY VISORP.

00170  0621  CE FFAD  INITAL LDX   #$FFAD        ESTABLISH B SIDE OF INTERRUPTS/LEDS & BEEP INTERFAC
00171  0624  FF 8012         STX   MAXOUT         AS OUTPUT AND INITIALIZE CONTROL REGISTER OF B.
00172  0627  7F 8012         CLR   MAXOUT

00174  062A  86 FF           LDA A #$FF          ESTABLISH A & B SIDES OF RESPONSE/REINFORCEMENT
00175  062C  C6 04           LDA B #4              INTERFACE AS INPUT & OUTPUT, RESPECTIVELY,
00176  062E  F7 8019         STA B INRSPS+1        AND INITIALIZE CONTROL REGISTERS OF A & B.
00177  0631  B7 801A         STA A OUTRNF
00178  0634  F7 8018         STA B OUTRNF+1
00179  0637  B7 801A         STA A OUTRNF

00181  063A  39             RTS                 RETURN TO MAIN PROGRAM

00183                    ******************************** END OF INITAL SR *********************************
```

```
00185                    ********************************************************************************
00186                    *                              CLEAR SR                                        *
00187                    ********************************************************************************

00189                    *    THIS SR IS CALLED BY VISCRP MAIN PROGRAM PRIOR TO EACH SESSION TO CLEAR
00190                    *  OUT THE SECTION OF LOWER MEMORY THAT IT USES FOR VARIABLES AND FLAGS.
00191                    *  SEVERAL VARIABLES ARE INITIALIZED AND DEFAULTS ARE ASSIGNED HERE.

00193   063B   CE  0000 CLEAR  LDX   #0

00195   063E   6F  00   EVRCLR CLR   0,X             CLEAR MEMORY LOCATIONS 0000 TO 009F
00196   0640   08             INX
00197   0641   8C  00A0       CPX   #$00A0
00198   0644   26  F8         BNE   EVRCLR

00200   0646   86  04         LDA A #UDFLT          NO. OF UNITS DEFAULT: 4
00201   0648   97  67         STA A UNITS

00203   064A   CE  03FF       LDX   #TBLSIZ-1       INIT. SAMPLE TABLE'S POINTR
00204   064D   DF  69         STX   POINTR

00206   064F   CE  520D       LDX   #FIRST2         INITIALIZE RNDSGN WITH $520DC8B9
00207   0652   DF  76         STX   RNDSGN
00208   0654   CE  C8B9       LDX   #SECND2
00209   0657   DF  78         STX   RNDSGN+2

00211   0659   CE  0A6A       LDX   #INTRPT         STORE ADDRESS OF INTERRUPT SR
00212   065C   FF  A000       STX   IOV

00214   065F   86  1E         LDA A #STDFLT         SESSION TIME DEFAULT: 30 MINUTES
00215   0661   97  57         STA A STIME

00217   0663   86  14         LDA A #DIVISR         INITIALIZE SFRQDV WITH DIVISR
00218   0665   97  56         STA A SFRQDV

00220   0667   86  3C         LDA A #60             INITIALIZE MINUTE WITH 60
00221   0669   97  58         STA A MINUTE

00223   066B   39             RTS

00225                    ********************************** END OF CLEAR SR ******************************
```

```
00227                    ********************************************************************************
00228                    *                                INPUT SR                                      *
00229                    ********************************************************************************

00231                    *    THIS SR IS CALLED BY VISORP MAIN PROGRAM TO RECEIVE SESSION PARAMETERS FROM
00232                    *  THE TERMINAL PRIOR TO EACH SESSION.
00233                    *    EACH HORIZONTAL LINE OF INPUT IS BEGUN BY ENTERING A SINGLE IDENTIFYING
00234                    *  CHARACTER PRECEDED BY ANY NUMBER OF SPACES. THE FUNCTION OF EACH OF THESE
00235                    *  SPECIAL CHARACTERS IS DEFINED BELOW.
00236                    *
00237                    *      '#' INDICATES THAT THE FOLLOWING DIGIT IS THE MAXIMUM NUMBER OF UNITS TO
00238                    *    BE SERVICED DURING THE SESSION. MUST BE THE FIRST LINE OF INPUT PRIOR TO
00239                    *    EACH SESSION IF USED. IF NOT, THEN THE MAXIMUM # OF UNITS DEFAULTS TO 4
00240                    *    AND SUBSEQUENT ENTERING OF'#' RESULTS IN ERROR CONDITION. (EX. #  7)
00241                    *      UNIT NO. INDICATES WITH WHICH UNIT THE FOLLOWING PARAMETERS ARE TO BE
00242                    *    ASSOCIATED. UNIT NO. MUST BE BETWEEN 1 AND THE MAXIMUM NUMBER OF UNITS
00243                    *    TO BE SERVICED. FOLLOW WITH AT LEAST ONE SPACE. ON THE SAME LINE, ENTER
00244                    *    THE UNIT'S VI (MEAN TIME INTERVAL), SD (TIME FROM VI AT WHICH 1 STANDARD
00245                    *    DEVIATION IS TO OCCUR), AND REINMX (MAXIMUM NUMBER OF REINFORCEMENTS TO
00246                    *    BE GIVEN THE ANIMAL). SEPARATE EACH OF THE ABOVE THREE ITEMS WITH AT LEAST
00247                    *    ONE SPACE. FOLLOW THE REINMX NUMBER WITH A SPACE. RANGE OF PARAMETERS:
00248                    *          VI       0 -> 255
00249                    *          SD       0 -> 85
00250                    *       REINMX      0 -> 999
00251                    *    IF 0 IS ENTERED FOR REINMX THEN A LARGE NUMBER OF REINFORCEMENTS (65535)
00252                    *    WILL BE AVAILABLE TO THE ANIMAL. IF INVALID CHARACTERS OR VALUES ARE
00253                    *    ENTERED FOR ANY OF THE PARAMETERS, THE WHOLE LINE IS REJECTED AND MUST BE
00254                    *    REENTERED BEFORE STARTING SESSION. (EX. 2 32 15 500)
00255                    *      'T' THE NUMBER FOLLOWING 'T', AFTER ANY NUMBER OF SPACES, IS THE SESSION
00256                    *    TIME LIMIT IN MINUTES. MUST BE BETWEEN 1 AND 255 AND FOLLOWED BY A SPACE.
00257                    *    IF NOT ENTERED, SESSION TIME DEFAULT IS 30 MINUTES. (EX. T   120)
00258                    *      'S' A REQUEST BY THE USER TO START THE SESSION. IF ALL PARAMETERS FOR ALL
00259                    *    OF THE UNITS TO BE SERVICED HAVE BEEN DEFINED, THEN TERMINAL RESPONDS BY
00260                    *    PRINTING 'TART' AFTER 'S'-SPELLING 'START' AND THE SESSION BEGINS. IF
00261                    *    SOMETHING IS AMISS (SUCH AS UNDEFINED UNIT PARAMETERS), THEN A QUESTION
00262                    *    MARK IS PRINTED. THE USER SHOULD CORRECT THE PROBLEM AND REQUEST THE
00263                    *    SESSION TO START AGAIN BY ENTERING 'S'.
00264                    *      'X' JUMP TO COMPUTER SYSTEM'S MONITOR.
00265                    *      $5E (HAT OR INVERTED 'V') RESTART ENTIRE INPUTTING PHASE: NEGLECT ALL
00266                    *    PREVIOUSLY ENTERED PARAMETERS AND START OVER.
00267                    *
00268                    *    UNIT NO.'S DO NOT HAVE TO BE IN ORDER AS IN:1 30  15 100
00269                    *                                               2  0   0   0
00270                    *                                               3 10   5 500
00271                    *                                               4 60  30 360
00272                    *  BOTH UNIT NO.'S AND 'T' MAY BE REENTERED TO REDEFINE PARAMETERS-THE LAST
00273                    *  PARAMETERS ENTERED BEFORE THE SESSION STARTS ARE THE ONES THAT ARE USED.
00274                    *    REMEMBER, ONLY UNIT NO.'S AND 'S' ARE REQUIRED DURING INPUTTING PHASE.

00276  066C  CE  04D0  INPUT  LDX   #MSG1          PRINT CR,5(LF),'VISORP',CR,2(LF)
00277  066F  BD  E07E         JSR   PDATA1

00279                    *  FIRST LINE'S FIRST PARAMETER INPUT    UNIT#, '#', 'T', 'S', 'X', $5E (INV 'V'

00281  0672  BD  0786  TRYAGN JSR   SINEEE          PLACE WHERE OVERRIDING # OF UNITS IS IN. BY USER
```

```
00282  0675  81 23        CMP A #'#
00283  0677  26 06        BNE   FIRSTP
00284  0679  7E 0756      JMP   POUNDS

00286                    *  FIRST PARAMETER INPUT    UNIT#, 'T','S','X',$5E (INV 'V')

00288  067C  BD 0786  INUNIT JSR   SINEEE          WAIT FOR ASCII -= $20 (SPACES)
00289  067F  81 53    FIRSTP CMP A #'S
00290  0681  26 03        BNE   NOTS
00291  0683  7E 07CB      JMP   START             IF CHAR. = 'S', JMP TO START

00293  0686  81 54    NOTS   CMP A #'T
00294  0688  26 03        BNE   NOTT
00295  068A  7E 0770      JMP   TPARAM            IF 'T', BRA TO SR TO CHANGE SESSION TIME

00297  068D  81 5E    NOTT   CMP A #$5E           IF $5E (INVERTED V), JMP SESHUN+2 (RESTART INPUT)
00298  068F  26 05        BNE   NOT5E
00299  0691  31           INS
00300  0692  31           INS
00301  0693  7E 0604      JMP   SESHUN+2

00303  0696  81 58    NOT5E  CMP A #'X            IF 'X', JMP TO MONITOR (GET OUT OF VISORP)
00304  0698  26 03        BNE   NOTX
00305  069A  7E E0E3      JMP   MONITR

00307  069D  81 31    NOTX   CMP A #'1            IF CHAR. < '1' OR CHAR. > UNITS, JMP HUH1
00308  069F  2C 02        BGE   OK2
00309  06A1  20 45        BRA   COMAGN

00311  06A3  80 30    CK2    SUB A #'0
00312  06A5  91 67        CMP A UNITS
00313  06A7  2E F8        BGT   NOTX+4
00314  06A9  97 5E        STA A OFFSET+1
00315  06AB  7A 005E      DEC   OFFSET+1

00317  06AE  5F           CLR B
00318  06AF  0D           SEC
00319  06B0  59        SHIFT1 ROL B               OBTAIN UNIT'S BIT POSITION IN B
00320  06B1  4A           DEC A
00321  06B2  26 FC        BNE   SHIFT1
00322  06B4  53           COM B
00323  06B5  D4 53        AND B ALLTST
00324  06B7  D7 53        STA B ALLTST            CLEAR UNIT'S BIT POSITION IN ALLTST

00326  06B9  BD E1AC      JSR   INEEE             MUST INPUT AT LEAST 1 SPACE
00327  06BC  81 20        CMP A #$20
00328  06BE  26 28        BNE   COMAGN              IF NOT, BRA COMAGN

00330                    *  VI INPUT    0 <= VI <= 255

00332  06C0  C6 04        LDA B #4                B <- MAX. DIGITS ALLOWED + 1
00333  06C2  BD 0794      JSR   NOLDGO            STORE INPUT IN INTERM BUFFER

00335  06C5  E6 6E        LDA B INTERM,X          GET 1'S DIGIT FROM INTERM BUFFER
00336  06C7  A6 6D        LDA A INTERM-1,X        GET 10'S DIGIT FROM INTERM BUFFER
00337  06C9  DF 5F        STX   SAVE#
```

```
00338  06CB  DE 5D          LDX   OFFSET
00339  06CD  E7 00          STA B VI,X          STORE 1'S DIGIT IN VI TABLE

00341  06CF  BD 07C5        JSR   TIME10        MULTIPLY 10'S DIGIT BY 10
00342  06D2  AB 00          ADD A VI,X          ADD TO VI TABLE
00343  06D4  A7 00          STA A VI,X

00345  06D6  DE 5F          LDX   SAVEX
00346  06D8  5F             CLR B               B <- 0
00347  06D9  A6 6C          LDA A INTERM-2,X    GET 100'S DIGIT FROM INTERM BUFFER
00348  06DB  27 0E          BEQ   OK1           IF = 0, BRA OK1
00349  06DD  C6 64          LDA B #100          B <- 100
00350  06DF  81 01          CMP A #1
00351  06E1  27 08          BEQ   OK1           IF = 1, BRA OK1
00352  06E3  58             ASL B               B <- B * 2
00353  06E4  81 02          CMP A #2
00354  06E6  27 03          BEQ   OK1           IF = 2,BRA OK1
00355  06E8  7E 07C0  COMAGN JMP   HUH1          IF <= 0, <= 1, OR <= 2, JMP HUH1

00357  06EB  DE 5D    OK1   LDX   OFFSET
00358  06ED  EB 00          ADD B VI,X          ADD TO VI TABLE
00359  06EF  25 F7          BCS   COMAGN        IF VI > 255, BRA HUH1
00360  06F1  E7 00          STA B VI,X          IF VI <= 255, VI <- B

00362                  *  SD INPUT    0<= SD <= 85

00364  06F3  C6 03          LDA B #3            B <- MAX. DIGITS ALLOWED + 1
00365  06F5  BD 0794        JSR   NOLDGO        STORE INPUT IN INTERM BUFFER

00367  06F8  E6 6E          LDA B INTERM,X      GET 1'S DIGIT FROM INTERM BUFFER
00368  06FA  A6 6D          LDA A INTERM-1,X    GET 10'S DIGIT FROM INTERM BUFFER
00369  06FC  DE 5D          LDX   OFFSET
00370  06FE  E7 08          STA B SD,X          STORE 1'S DIGIT IN SD TABLE

00372  0700  BD 07C5        JSR   TIME10        MULTIPLY 10'S DIGIT BY 10
00373  0703  AB 08          ADD A SD,X          ADD TO SD TABLE
00374  0705  81 55          CMP A #85           IF SD > 85, JMP HUH1
00375  0707  22 DF          BHI   COMAGN
00376  0709  A7 08          STA A SD,X          STORE IN SD TABLE

00378                  *  REINMX(REINFORCEMENT MAXIMUM) INPUT    0 <= REINMX <= 999

00380  070B  78 005E        ASL   OFFSET+1      (OFFSET+1) <- (OFFSET+1) * 2

00382  070E  C6 04          LDA B #4            B <- MAX. DIGITS ALLOWED + 1
00383  0710  BD 0794        JSR   NOLDGO        STORE INPUT IN INTERM BUFFER

00385  0713  E6 6E          LDA B INTERM,X      GET 1'S DIGIT FROM INTERM BUFFER
00386  0715  A6 6D          LDA A INTERM-1,X    GET 10'S DIGIT FROM INTERM BUFFER
00387  0717  DF 5F          STX   SAVEX
00388  0719  DE 5D          LDX   OFFSET
00389  071B  E7 11          STA B REINMX+1,X    STORE 1'S DIGIT IN REINMX TABLE

00391  071D  BD 07C5        JSR   TIME10        MULTIPLY 10'S DIGIT BY 10
00392  0720  AB 11          ADD A REINMX+1,X    ADD TO REINMX TABLE
00393  0722  A7 11          STA A REINMX+1,X
```

```
00395  0724  DE  5F          LDX   SAVEX
00396  0726  A6  6C          LDA A INTERM-2,X      GET 100'S DIGIT FROM INTERM BUFFER
00397  0728  DE  5D          LDX   OFFSET

00399  072A  BD  07C5        JSR   TIME10          MSB(B),LSB(A) <- A * 100
00400  C72D  5F              CLR B
00401  072E  A7  10          STA A REINMX,X
00402  0730  48              ASL A
00403  C731  48              ASL A
00404  0732  59              ROL B
00405  0733  AB  10          ADD A REINMX,X
00406  0735  48              ASL A
00407  0736  59              ROL B
00408  0737  AB  11          ADD A REINMX+1,X      ADD TO REINMX TABLE
00409  0739  A7  11          STA A REINMX+1,X
00410  073B  C9  00          ADC B #0
00411  073D  E7  10          STA B REINMX,X


00413  073F  96  5E          LDA A OFFSET+1
00414  0741  44              LSR A
00415  0742  4C              INC A
00416  0743  5F              CLR B
00417  0744  0D              SEC
00418  0745  59        SHIFT2 ROL B               OBTAIN UNIT'S BIT POSITION
00419  0746  4A              DEC A
00420  0747  26  FC          BNE   SHIFT2
00421  0749  DA  53          ORA B ALLTST          SET UNIT'S BIT POSITION IN ALLTST
00422  C74B  D7  53          STA B ALLTST

00424  074D  CE  052A  NEWLIN LDX   #CRLF          SKIP TO BEGINNING OF NEXT LINE
00425  C75C  BD  E07E        JSR   PDATA1

00427  0753  7E  067C        JMP   INUNIT          BRANCH BACK FOR ANOTHER UNIT FIELD


00429                  ************************** END OF INPUT SR ***********************************
```

```
00431                    ********************************************************************************
00432                    *                              POUNDS SR                                       *
00433                    ********************************************************************************

00435                    *    THIS CODE SEGMENT IS CALLED BY INPUT SR TO RECEIVE THE MAXIMUM NUMBER OF
00436                    *  UNITS TO BE SERVICED DURING A SESSION. THIS NUMBER CAN BE FROM 1 TO 8 AND CAN
00437                    *  BE PRECEDED BY ANY NUMBER OF SPACES. AFTER ENTERING THE DIGIT, THE SR RETURNS
00438                    *  TO INPUT SR. IF A NON-NUMERIC CHARACTER OR AN OUT OF RANGE NUMBER IS ENTERED,
00439                    *  THEN A QUESTION MARK IS PRINTED AND SR RETURNS TO INPUT SR. IF POUNDS SR IS
00440                    *  NOT CALLED, THEN, BY DEFAULT, 4 UNITS WILL BE SERVICED.

00442    0756    BD  0786  POUNDS JSR   SINEEE          WAIT FOR ASCII >= $20
00443    0759    81  31          CMP A #'1              IF CHAR. < '1' OR CHAR. > '8', BRA INVALD
00444    075B    2D  04          BLT   INVALD           IF '1' <= CHAR. <= '8', BRA SUPERC
00445    075D    81  38          CMP A #'8
00446    075F    2F  09          BLE   SUPERC

00448    0761    CE  0529  INVALD LDX   #QUEST          PRINT '?', CR, LF
00449    0764    BD  E07E         JSR   POATA1
00450    0767    7E  0672         JMP   TRYAGN          LOOP BACK FOR ANOTHER CHAR.

00452    076A    80  30    SUPERC SUB A #'0             A <- A - $30
00453    076C    97  67           STA A UNITS           A SUPERCEDES UNIT # DEFAULT OF 4

00455    076E    20  DD           BRA   NEWLIN          RETURN TO INPUT SR

00457                    ************************** END OF POUNDS SR ************************************
```

```
00459                    ********************************************************************************
00460                    *                              TPARAM SR                                       *
00461                    ********************************************************************************

00463                    *    THIS CODE SEGMENT IS CALLED BY INPUT SR TO RECEIVE THE SESSION TIME LIMIT
00464                    *  FROM THE TERMINAL. IF NOT CALLED, THE SESSION WILL LAST 30 MINUTES BY
00465                    *  DEFAULT. THE USER CAN ENTER ANY NUMBER OF SPACES BEFORE ENTERING THE SESSION
00466                    *  TIME. THE SESSION TIME CAN BE FROM 1 MINUTE TO 255 MINUTES LONG, ONLY IN 1
00467                    *  MINUTE INTERVALS. AFTER ENTERING DIGIT(S), ENTERING A SPACE CAUSES CONTROL
00468                    *  TRANSFER TO INPUT SR. IF INVALID CHARACTERS (NOT NUMERIC OR SPACE OR OUT OF
00469                    *  RANGE NUMBERS) ARE ENTERED, A QUESTION MARK IS PRINTED AND SR TRANSFERS TO
00470                    *  INPUT SR, LEAVING PREVIOUS VALUE OF STIME UNCHANGED.

00472                    *  SESSION TIME PARAMETER INPUT (OVERRIDE DEFAULT: 30)     0 <  STIME <= 255

00474  0770  C6  04  TPARAM LDA B #4              B <- MAX. DIGITS ALLOWED + 1
00475  0772  8D  20        BSR   NOLOGO           STORE INPUT IN INTERM BUFFER

00477  0774  A6  6D        LDA A INTERM-1,X       GET 10'S DIGIT FROM INTERM BUFFER
00478  0776  8D  4D        BSR   TIME10           MULTIPLY BY 10
00479  0778  AB  6E        ADD A INTERM,X         ADD 1'S DIGIT FROM INTERM BUFFER TO TEMPA
00480  077A  97  74        STA A TEMPA

00482  077C  5F            CLR B                  B <- 0
00483  077D  A6  6C        LDA A INTERM-2,X       GET 100'S DIGIT FROM INTERM BUFFER
00484  077F  27  08        BEQ   OK               IF = 0, BRA OK
00485  0781  C6  64        LDA B #100             B <- 100
00486  0783  81  01        CMP A #1
00487  0785  27  05        BEQ   OK               IF = 1, BRA OK
00488  0787  58            ASL B                  B <- B * 2
00489  0788  81  02        CMP A #2
00490  078A  26  34        BNE   HUH1             IF ~= 0, ~= 1, OR ~= 2, BRA HUH1

00492  078C  DB  74  OK    ADD B TEMPA            ADD TEMPA TO B
00493  078E  23  30        BLS   HUH1             IF = 0 OR > 255, BRA HUH1
00494  0790  D7  57        STA B STIME              OTHERWISE, STIME <- B

00496  0792  20  89        BRA   NEWLIN           (AN ENTRY POINT IN INPUT SR)

00498                    ************************** END OF TPARAM SR **********************************
```

```
00500                    ********************************************************************************
00501                    *                                  NOLDGO SR                                   *
00502                    ********************************************************************************

00504                    *    THIS SR IS CALLED BY INPUT SR AND TPARAM SR TO RECEIVE DATA FROM THE
00505                    *  TERMINAL AND STORE IN INTERM TABLE. B IS PASSED INTO THIS SR AS THE MAXIMUM
00506                    *  NUMBER OF DIGITS THAT NOLDGO IS TO ALLOW TO BE ENTERED FROM TERMINAL. IF TOO
00507                    *  MANY CHARACTERS, THEN PRINT A QUESTION MARK AND RETURN TO INPUT SR. ONLY
00508                    *  NUMERIC CHARACTERS (0 -> 9) ARE VALID. IF ANY INVALID CHARACTERS, THEN PRINT
00509                    *  A QUESTION MARK AND RETURN TO INPUT SR. ENTERRING A SPACE AFTER THE DIGIT(S)
00510                    *  CAUSES RETURN FROM SR. REGISTERS A & B ARE DESTROYED. REGISTER X IS PASSED
00511                    *  OUT OF SR WITH THE OFFSET IN INTERM TABLE WHERE THE LAST DIGIT ENTERRED IS
00512                    *  STORED. THE PURPOSE OF THIS SR IS SO THAT LEADING ZEROS IN A NUMBER DO NOT
00513                    *  HAVE TO BE ENTERRED AT THE TERMINAL, RESULTING IN AN EASIER, FREE/FORMAT WAY
00514                    *  OF ENTERRING DATA.

00516    C794    4F          NOLDGO CLR A
00517    C795    97  6C             STA A INTERM-2
00518    C797    97  6D             STA A INTERM-1
00519    C799    CE  FFFF           LDX   #$FFFF
00520    C79C    8D  18             BSR   SINEEE          WAIT FOR ASCII -= $20

00522    C79E    81  30     ENTER   CMP A #'0             TEST IF '0' <= A <= '9'
00523    C7A0    2D  1C             BLT   HUH2              FALSE, BRA HUH2
00524    C7A2    81  39             CMP A #'9               TRUE, CONTINUE
00525    C7A4    2E  18             BGT   HUH2
00526    C7A6    80  30             SUB A #'0             A <- A - $30
00527    C7A8    08                 INX
00528    C7A9    5A                 DEC B                 IF TOO MANY CHARS. INPUTTED, BRA HUH2
00529    C7AA    27  12             BEQ   HUH2
00530    C7AC    A7  6E             STA A INTERM,X        STORE CHAR. IN INTERM BUFFER
00531    C7AE    BD  E1AC           JSR   INEEE           INPUT NEXT CHAR.
00532    C7B1    81  20             CMP A #$20            IF NOT A SPACE,
00533    C7B3    26  E9             BNE   ENTER             CONTINUE ENTERING DIGITS

00535    C7B5    39                 RTS                   RETURN FROM SR


00537                    ******************************** END OF NOLDGO SR **********************************
```

```
00539                    ********************************************************************************
00540                    *                                SINEEE SR                                     *
00541                    ********************************************************************************

00543                    *    THIS SR IS CALLED BY INPUT SR, POUNDS SR, NOLDGO SR, AND ASK SR TO RETURN
00544                    *  IN REGISTER A, AN ASCII CHARACTER FROM THE TERMINAL IF NOT A SPACE.

00546  07B6  BD  E1AC  SINEEE JSR   INEEE          RETURN CHAR. IN REG. A WHEN NOT A SPACE ($20)
00547  07B9  81  20          CMP A #$20            OTHERWISE, WAIT FOR NON-SPACE CHAR.
00548  07BB  27  F9          BEQ   SINEEE          (ALLOWS SPACING BETWEEN INPUT ITEMS)

00550  07BD  39              RTS                   RETURN FROM SR

00552                    ****************************** END OF SINEEE SR ********************************
```

```
00554                    ********************************************************************************
00555                    *                                HUH172 SR                                     *
00556                    ********************************************************************************

00558                    *    THIS CODE SEGMENT IS CALLED BY START SR, NOLDGO SR, TPARAM SR, AND INPUT SR
00559                    *  TO PRINT A QUESTION MARK AS AN ALL-PURPOSE ERROR MESSAGE AND THEN REENTER
00560                    *  INPUT SR.

00562   07BE   31        HUH2   INS                    ENTRY POINT IF FROM SR
00563   07BF   31               INS
00564   07C0   CE  0529  HUH1   LDX   #QUEST           PRINT '?', CR, LF

00566   07C3   20  8B           BRA   NEWLIN+3

00568                    ********************************** END OF HUH172 SR ****************************
```

```
00570                    ********************************************************************************
00571                    *                                  TIME10 SR                                   *
00572                    ********************************************************************************

00574                    *    THIS SR IS CALLED BY INPUT SR AND TPARAM SR TO MULTIPLY REGISTER A BY 10
00575                    *  AND RETURN THE RESULT IN REGISTER A. REGISTER B IS DESTROYED.

00577    07C5    16      TIME10 TAB                  RETURNS A <- A * 10
00578    07C6    48             ASL A
00579    07C7    48             ASL A
00580    07C8    1B             ABA
00581    07C9    48             ASL A

00583    07CA    39             RTS

00585                    ******************************** END OF TIME10 SR ********************************
```

```
00587                          ********************************************************************************
00588                          *                                START SR                                      *
00589                          ********************************************************************************

00591                          *    THIS SR IS CALLED BY INPUT SR TO START A SESSION IF ALL SESSION PARAMETERS
00592                          *  HAVE BEEN DEFINED.
00593                          *    ALLTST IS PASSED INTO THIS SR TO INDICATE IF ALL SESSION PARAMETERS HAVE
00594                          *  BEEN DEFINED. A UNIT'S BIT POSITION IN ALLTST IS HIGH IF ALL PARAMETERS
00595                          *  ASSOCIATED WITH THAT UNIT (VI, SD, & REINMX) HAVE BEEN ENTERED. IT IS LOW IF
00596                          *  NONE OF THE PARAMETERS HAVE BEEN DEFINED OR IF AN ERROR OCCURED WHILE
00597                          *  ENTERING ANY OF THE THREE PARAMETERS. FOR EXAMPLE, IF THREE UNITS ARE TO BE
00598                          *  SERVICED BY VISORP AND ALLTST CONTAINS 0000 0111, THEN START SR WOULD SET UP
00599                          *  A NEW SESSION AND ALLOW IT TO BEGIN. IF ALLTST CONTAINED 0000 0101, THEN
00600                          *  START SR WOULD PRINT A QUESTION MARK & RETURN TO INPUT SR TO AWAIT COMPLETION
00601                          *  OF INPUT PHASE. THIS WAY, A SESSION CAN NOT BEGIN WITH UNDEFINED OR ERRONEOUS
00602                          *  PARAMETERS WHICH MIGHT CAUSE UNPREDICTABLE RESULTS.
00603                          *    IF ALL SESSION PARAMETERS ARE DEFINED, THEN UNITSP IS INITIALIZED
00604                          *  ACCORDING TO UNITS (SEE BELOW), SUCCESSFUL START IS INDICATED TO USER BY
00605                          *  PRINTING OUT 'TART' AFTER USER'S 'S', THE UNITS' ELEMENTS IN CLOCK AND FREQDV
00606                          *  TABLES ARE INITIALIZED, AND SR RETURNS TO VISORP.


00608  07CB  D6 53   START  LDA B ALLTST        HAS ALL DATA BEEN ENTERED FOR ALL UNITS?
00609  07CD  96 67          LDA A UNITS
00610  07CF  4C             INC A
00611  07D0  4A      SHIFTO DEC A
00612  07D1  54             LSR B
00613  07D2  25 FC          BCS   SHIFTO
00614  07D4  4D             TST A
00615  07D5  26 E9          BNE   HUH1          IF NO, BRA HUH1

00617  07D7  96 67          LDA A UNITS         COMPUTE BIT POSITION IN UNITSP BY # OF UNITS
00618  07D9  5F             CLR B                 (I.E. IF UNITS=3 THEN UNITSP=00000100)
00619  07DA  0D             SEC

00621  07DB  59      SHIFTB ROL B
00622  07DC  4A             DEC A
00623  07DD  26 FC          BNE   SHIFTB
00624  07DF  D7 68          STA B UNITSP

00626  07E1  CE 04E0        LDX   #MSG2         PRINT 'TART', CR, LF, LF
00627  07E4  BD E07E        JSR   PDATA1
00628  07E7  8D 01          BSR   MOVE1         INITIALIZE UNIT'S CLOCKS & FREQDV'S

00630  07E9  39             RTS                 RETURN TO MAIN PROGRAM

00632                          ****************************** END OF START SR ***********************************
```

```
00634                    ********************************************************************************
00635                    *                                MOVE1 SR                                      *
00636                    ********************************************************************************

00638                    *    THIS SR IS CALLED AT THE BEGINNING OF A SESSION BY START SR TO INITIALIZE
00639                    *  THE UNITS' CLOCKS WITH VARIABLE TIME INTERVALS AND TO INITIALIZE THE UNITS'
00640                    *  ELEMENTS IN FREQDV TABLE WITH DIVISR (20). REGISTERS A, B, & X ARE DESTROYED.


00642  07EA  CE  0000  MOVE1  LDX   #0

00644  07ED  A6  00           LDA A VI,X          A <- UNIT'S VI
00645  07EF  E6  08           LDA B SD,X          B <- UNIT'S SD
00646  07F1  8D  0C           BSR   DEVIAT        A <- ADJUSTED SAMPLE FROM DEVIAT SR
00647  07F3  A7  40           STA A CLOCK,X       M(CLOCK) <- A

00649  07F5  86  14           LDA A #DIVISR       INITIALIZE FREQDV'S WITH DIVISR TO BE COUNTED
00650  07F7  A7  48           STA A FREQDV,X        DOWN TO ZERO

00652  07F9  08              INX
00653  07FA  9C  66           CPX   UNITS-1       ALL UNITS HAVE NORM. VALUES & DIVISR'S?
00654  07FC  26  EF           BNE   MOVE1+3         IF NO, DO ANOTHER UNIT

00656  07FE  39              RTS                   IF YES, RETURN

00658                    ********************************* END OF MOVE1 SR *********************************
```

```
00660                    ********************************************************************************
00661                    *                              DEVIAT SR                                       *
00662                    ********************************************************************************

00664                    *    THIS SR IS CALLED BY MOVEI SR AND RCREIN SR TO PROVIDE NORMALLY DISTRIBUTED
00665                    *  VALUES ABOUT A GIVEN MEAN AND STANDARD DEVIATION TO BE USED IN UNIT "CLOCKS"
00666                    *  AS VARIABLE TIME INTERVALS. THIS SR REQUIRES THE STATIC DATA BASE: SAMPLE
00667                    *  (1024 BYTES) AND THE 4-BYTE DATA BASE: RNDSGN TO FUNCTION.
00668                    *    THE MEAN (VI) IS PASSED INTO THIS SR THROUGH REGISTER A AND THE DISPERSION
00669                    *  (SD: THE TIME OUT FROM THE MEAN AT WHICH 1 STANDARD DEVIATION OCCURS) IS
00670                    *  PASSED INTO THIS SR THROUGH REGISTER B. THE NORMALLY DISTRIBUTED RESULT IS
00671                    *  RETURNED IN REGISTER A. B IS DESTROYED, BUT X IS NOT. A MORE THOROUGH
00672                    *  DISCUSSION OF THIS METHOD OF OBTAINING VARIABLE INTEGERS CAN BE FOUND IN THE
00673                    *  PAPER: FUNCTIONAL DESCRIPTION OF THE VI SCHEDULE OF REINFORCEMENT PROGRAM.


00675  07FF  97 7A  DEVIAT STA A SAVVI          SAVE UNIT'S VI
00676  0801  DF 5F         STX   SAVEX          SAVEX <- X

00678  0803  DE 69         LDX   POINTR
00679  0805  A6 A0         LDA A SAMPLE,X       A <- NUMBER FROM SAMPLE TABLE AT POINTR OFFSET
00680  0807  09            DEX                  DECREMENT POINTR VALUE
00681  0808  8C FFFF       CPX   #$FFFF         IF NO ROLL-OVER FROM ZERO, THEN BRANCH NOTRST
00682  080B  26 03         BNE   NOTRST
00683  080D  CE 03FF       LDX   #TBLSIZ-1        ELSE REINITIALIZE POINTR W/ MAX. OFFSET
00684  0810  DF 69  NOTRST STX   POINTR

00686  0812  DE 5F         LDX   SAVEX          X <- SAVEX

00688                    *    MULTIPLIES SAMPLE BY SD:    MSB(A),LSB(B) <- A * B

00690  0814  97 71         STA A MCND           STORE MULTIPLICAND (SAMPLE)
00691  0816  D7 72         STA B MPLR           STORE MULTIPLIER (SD)

00693  0818  86 F8         LDA A #$F8           MPYCNT <- -8
00694  081A  97 73         STA A MPYCNT

00696  081C  4F            CLR A                A,B <- 0
00697  081D  5F            CLR B

00699  081E  58  MPYRLP   ASL B                 A,B <- SHIFT LEFT(A,B)
00700  081F  49            ROL A

00702  0820  78 0072       ASL   MPLR           MPLR <- SHIFT LEFT(MPLR)
00703  0823  24 04         BCC   MPYNCR         CARRY?
00704  0825  DB 71         ADD B MCND           AB <- AB + MCND
00705  0827  89 00         ADC A #0

00707  0829  7C 0073  MPYNCR INC   MPYCNT       MPYCNT <- MPYCNT + 1
00708  082C  26 F0         BNE   MPYRLP         MPYCNT = 0 ?

00710                    *  MULTIPLIES A,B BY 3 :    (A,B) <- (A,B) * 3

00712  082E  97 74         STA A TEMPA          SAVE A,B IN TEMPA/B
00713  0830  D7 75         STA B TEMPB
```

```
00715  0832  58          ASL B              A,B <- A,B * 2
00716  0833  49          ROL A

00718  0834  DB 75       ADD B TEMPB        A,B <- (A,B) + (TEMPA,TEMPB)
00719  0836  99 74       ADC A TEMPA        A HAS (A,B)/256 ;  B IS NOT USED

00721                 *  RANDOM BITS IN RNDSGN DECIDE IF A IS TO BE SUB. OR ADD. FROM/TO VI

00723  0838  78 0079     ASL   RNDSGN+3      RNDSGN <- ROLL LEFT(RNDSGN)
00724  0838  79 0078     ROL   RNDSGN+2        (ROLLING BITS OUT INTO CARRY)
00725  083E  79 0077     ROL   RNDSGN+1
00726  0841  79 0076     ROL   RNDSGN
00727  0844  24 0E       BCC   ADD          C = 0 MEANS ADD, SO BRANCH TO ADD
00728  0846  C6 01       LDA B #%00000001   IF C = 1, THEN 'ROLL' CARRY AROUND TO RNDSGN+3
00729  0848  DA 79       ORA B RNDSGN+3
00730  084A  D7 79       STA B RNDSGN+3

00732  084C  16          TAB                SAVE A IN B
00733  084D  96 7A       LDA A SAVVI        A <- UNIT'S VI
00734  084F  10          SBA                STORE A-B IN A
00735  0850  24 08       BCC   ADJSTD       IF LESS THAN ZERO, ASSIGN ZERO TO A
00736  0852  4F          CLR A
00737  0853  39          RTS                RETURN

00739  0854  9B 7A  ADD  ADD A SAVVI         A <- A + UNIT'S VI

00741  0856  24 02       BCC   ADJSTD       IF GREATER THAN 255, THEN ASSIGN $FF TO A
00742  0858  86 FF       LDA A #$FF

00744  085A  39       ADJSTD RTS            RETURN

00746                 ********** END OF DEVIAT SR **********
```

```
00748                    ********************************************************************************
00749                    *                                  IO SR                                       *
00750                    ********************************************************************************

00752                    *    THIS SR IS CALLED BY VISORP MAIN PROGRAM TO DO THE ACTUAL MONITORING OF
00753                    *  INPUTS AND THE ACTUAL TRIGGERING OF REINFORCEMENTS.
00754                    *    THIS SR RETURNS TRURSP TO BE USED BY RSPCHK. TRURSP WILL CONTAIN THE
00755                    *  CURRENT "TRUE" RESPONSES AS DEFINED BELOW AFTER RETURN.
00756                    *    TRIGS IS PASSED INTO THIS SR BY RCREIN SR TELLING IO SR TO TRIGGER
00757                    *  REINFORCEMENTS.

00759                    *  SOFTWARE PERFORMS THESE HARDWARE FUNCTIONS:
00760                    *    1) DEBOUNCING OF INPUTS (RESPONSES).
00761                    *       RESPONSE DEFINED AS LASTING > 50 MS.
00762                    *       BOUNCE OR NOISE DEFINED AS LASTING < 50 MS.
00763                    *
00764                    *    2) POINT OF RESPONSE DEFINED AS LOW TO HIGH TRANSITION (NORMAL LOGIC)

00766  C858  B6  8018  IO      LDA A INRSPS        GET RESPONSE DATA (INVERTED LOGIC)
00767  C85E  43          IO      COM A             CONVERT TO STANDARD LOGIC
00768  C85F  26  04              BNE   HAVDAT      IF SOME BITS SET, THEN CHECK INDIVIDUALLY
00769  0861  97  59              STA A PRVRSP      OTHERWISE STORE IN PRVRSP
00770  0863  20  0A              BRA   TRIGGR

00772  0865  16          HAVDAT TAB                REG.B <- REG.A
00773  0866  73  0059            COM   PRVRSP      PRVRSP <- 1'S COMPL.(PRVRSP)
00774  C869  94  59              AND A PRVRSP      A <- A & PRVRSP
00775  086B  D7  59              STA B PRVRSP      PRVRSP <- B  (PRVRSP GETS NEW VALUE)
00776  086D  97  54              STA A TRURSP      TRURSP <- A  (TRURSP GETS 'TRUE' RESPONSES)

00778                    *    3) "ONE-SHOT" LOGIC FOR TRIGGERING REINFORCEMENTS.
00779                    *       SOLENOIDS KEPT ON FOR 1 CLOCK PULSE (50 MS.)
00780                    *
00781                    *    3) BUFFERED TRIGGERING SO THAT NO MORE THAN 1 SOLENOID IS ON AT A TIME.
00782                    *       IF SEVERAL SOLENOIDS WERE TRIGGERED AT THE SAME TIME, THERE WOULD HAVE
00783                    *       BEEN AS MUCH AS 14 AMPS DRAWN FROM THE EXTERNAL POWER SUPPLY. THIS
00784                    *       BUFFERING PREVENTS BLOWING THE POWER SUPPLY'S FUSE.

00786  086F  86  FF    TRIGGR LDA A #$FF          TURN OFF ALL SOLENOIDS
00787  C871  B7  801A          STA A OUTRNF
00788  0874  96  6B            LDA A TRIGS       IF NO TRIGGERING IS NEEDED, BRA EXIT
00789  C876  27  11            BEQ   EXIT

00791  0878  86  01            LDA A #1          A <- 00000001

00793  087A  95  6B    LOOP    BIT A TRIGS       TEST BIT(A) IN TRIGS
00794  087C  26  03            BNE   BIT         IF SET, BRA BIT
00795  C87E  48                ASL A             OTHERWISE, A <- SHIFT LEFT(A)
00796  087F  20  F9            BRA   LOOP        BRANCH BACK FOR FURTHER BIT TESTS

00798  C881  43          BIT     COM A
00799  0882  B7  801A            STA A OUTRNF      TURN ON JUST ONE SOLENOID ACCORDING TO A
00800  0885  94  6B              AND A TRIGS       CLEAR THAT BIT IN TRIGS
00801  C887  97  6B              STA A TRIGS
```

```
00803    0889    39        EXIT   RTS                RETURN TO MAIN PROGRAM

00805    ********************************** END OF IO SR **********************************
```

```
00807                    ********************************************************************************
00808                    *                                  RSPCHK SR                                   *
00809                    ********************************************************************************

00811                    *    THIS SR IS CALLED BY VISORP MAIN PROGRAM TO RECORD RESPONSES AND MONITOR
00812                    *  UNITS WHERE VI = 0 (CONTINUOUS REINFORCEMENT).
00813                    *    TRURSP IS TESTED FOR ANY HIGH BITS. IF NONE ARE HIGH (NO RESPONSES HAVE
00814                    *  OCCURED), THEN RETURN TO VISORP. IF ANY ARE HIGH, THEN TEST EACH BIT IN
00815                    *  TRURSP.
00816                    *    THE FOLLOWING IS PERFORMED FOR ALL UNITS THAT ARE TO BE SERVICED. IF THE
00817                    *  UNIT'S BIT POSITION IN TRURSP IS NOT SET (NO RESPONSE) THEN CHECK NEXT UNIT'S
00818                    *  BIT POSITION; OTHERWISE, THE ANIMAL HAS RESPONDED SO CONTINUE. IF THE UNIT'S
00819                    *  BIT POSITION IN RMXFLG IS SET (THE ANIMAL HAS RECIEVED ALL OF ITS ALLOTTED
00820                    *  REINFORCEMENTS, THEREFORE, NO NEED TO MONITOR ANY LONGER), THEN CHECK NEXT
00821                    *  UNIT'S BIT POSITION; OTHERWISE CONTINUE. IF THE UNIT'S ELEMENT IN VI TABLE IS
00822                    *  ZERO (CONTINUOUS REINFORCEMENT) THEN RECORD (AND ISSUE) A REINFORCEMENT HERE
00823                    *  INSTEAD OF IN RENCHK SR. IF THE UNIT'S ELEMENT IN VI TABLE IS NOT ZERO, THEN
00824                    *  RECORD THE RESPONSE BY INCREMENTING THE UNIT'S ELEMENT IN RESPS TABLE.
00825                    *    AFTER ALL UNITS HAVE BEEN SERVICED, CLEAR TRURSP AND RETURN TO VISORP.


00827  088A  96 54  RSPCHK LDA A TRURSP          ANY TRUE RESPONSES?
00828  088C  97 7C        STA A RSPFLG
00829  088E  27 2E        BEQ   NOTRU           IF NO, RETURN TO MAIN PROGRAM
00830                    *                       IF YES, CHECK EACH BIT

00832  0890  D6 68        LDA B UNITSP          B CONTAINS UNIT BIT POSITION
00833  0892  DE 66        LDX   UNITS-1         X <- # OF UNITS(FOR TABLE OFFSET)

00835  C894  09     TSTBIT DEX                   X <- X - 1
00836  0895  D5 54        BIT B TRURSP          TEST UNIT'S BIT POSITION
00837  0897  27 20        BEQ   NEXTI             IF NOT SET, CHECK NEXT BIT

00839  0899  D5 50        BIT B RMXFLG          IF REINF. MAX. HAS NOT BEEN REACHED, CONTINUE
00840  089B  26 1C        BNE   NEXTI           IF REINF. MAX. HAS BEEN REACHED, CHECK NEXT BIT
00841  C89D  A6 00        LDA A VI,X            VI = 0 ?
00842  C89F  26 03        BNE   SKIPB           IF NOT, SKIP NEXT SECTION OF CODE-TO SKIPB

00844  C8A1  BD 094C      JSR   RCREIN          IF YES, RECORD THE REINFORCEMENT

00846  C8A4  DF 5F  SKIPB  STX   SAVEX
00847  08A6  DF 5D        STX   OFFSET
00848  08A8  78 005E      ASL   OFFSET+1
00849  C8AB  DE 5D        LDX   OFFSET

00851  08AD  A6 21        LDA A RESPS+1,X       INCREMENT # OF RESPONSES
00852  C8AF  8B 01        ADD A #1                (RESPS IS 2 BYTES LONG)
00853  08B1  A7 21        STA A RESPS+1,X
00854  C8B3  24 02        BCC   SKIPA
00855  C8B5  6C 20        INC   RESPS,X
00856  08B7  DE 5F  SKIPA  LDX   SAVEX

00858  C8B9  54     NEXTI  LSR B                 B <- SHIFT RIGHT(B)
00859  08BA  26 D8        BNE   TSTBIT          IF MORE DATA, CHECK NEXT BIT
00860  08BC  D7 54        STA B TRURSP
```

```
00862   088E   39        NOTRU  RTS                    RETURN TO MAIN PROGRAM

00864                    ****************************** END OF RSPCHK SR ********************************
```

```
00866                    ********************************************************************************
00867                    *                                  RENCHK SR                                   *
00868                    ********************************************************************************

C0870                    *    THIS SUBROUTINE IS CALLED BY VISORP MAIN PROGRAM TO DETERMINE IF
00871                    *  REINFORCEMENTS SHOULD BE ISSUED.
00872                    *    FOR A REINFORCEMENT TO BE ISSUED, TWO CONDITIONS MUST BE MET: 1) A
00873                    *  REINFORCEMENT MUST BE AVAILABLE AND 2) A RESPONSE MUST HAVE JUST OCCURRED.
00874                    *  FIRST OF ALL, THE TWO BYTES THAT CONTAIN THIS INFORMATION: RENFLG & RSPFLG,
00875                    *  RESPECTIVELY, ARE TESTED TO SEE IF THEY CONTAIN ANY HIGH BITS AT ALL. IF
00876                    *  NEITHER OF THEM DO, THEN RETURN TO VISORP. OTHERWISE, IF BOTH HAVE SOME SET
00877                    *  BITS, THEN TEST EACH BIT POSITION IN BOTH FLAGS.
00878                    *    THE FOLLOWING IS DONE FOR ALL UNITS TO BE SERVICED. TEST UNIT'S BIT
00879                    *  POSITION IN RENFLG & RSPFLG. IF EITHER IS ZERO, THEN CHECK NEXT UNIT'S FLAGS.
00880                    *  OTHERWISE, IF BOTH BITS IN THE SAME POSITION ARE SET, THEN CLEAR THAT BIT IN
00881                    *  RENFLG (THE BIT IN RSPFLG WILL BE EFFECTIVELY CLEARED LATER BY RSPCHK) AND
00882                    *  REINITIALIZE THAT UNIT'S ELEMENT IN FREQDV WITH DIVISR (20) (AFTER THE
00883                    *  UNIT'S FIRST TIME INTERVAL OF THE SESSION ELLAPSES, THE FIRST INITIALIZATION
00884                    *  WITHIN A TIME INTERVAL OF FREQDV IS PERFORMED HERE. THE SUBSEQUENT
00885                    *  INITIALIZATIONS WITHIN EACH TIME INTERVAL ARE PERFORMED IN BIGBEN SR.)
00886                    *  THE UNIT'S REINFORCEMENT IS RECORDED BY CALLING RCREIN SR.  DEVIAT SR IS
00887                    *  CALLED TO OBTAIN A NEW TIME INTERVAL TO STORE IN THE UNIT'S ELEMENT IN CLOCK.
00888                    *  THE UNIT'S VI (MEAN) AND SD (STANDARD DEVIATION) ARE RETRIEVED FROM THE VI
00889                    *  & SD TABLES AND PASSED INTO DEVIAT SR THROUGH THE A & B REGISTERS,
00890                    *  RESPECTIVELY. THE NEW TIME INTERVAL IS RETURNED THROUGH REGISTER A.
00891                    *    AFTER ALL UNITS HAVE BEEN SERVICED, RETURN TO VISORP.

00893  08BF  96 52  RENCHK LDA A RENFLG          ARE REINFORCEMENTS AVAILABLE?
00894  08C1  27 2D         BEQ   ALL            IF NO, RETURN TO MAIN PROGRAM
00895                    *                       IF YES, HAVE RESPONSES OCCURRED?
00896  08C3  96 7C         LDA A RSPFLG                 IF NO, RETURN TO MAIN PROGRAM
00897  08C5  27 29         BEQ   ALL                    IF YES, CHECK EACH BIT

00899  08C7  D6 68         LDA B UNITSP         B CONTAINS UNIT BIT POSITION
00900  08C9  DE 66         LDX   UNITS-1        X <- # OF UNITS(FOR TABLE OFFSET)

00902  08CB  09     LOOPA  DEX                  X <- X - 1
00903  08CC  D5 52         BIT B RENFLG         IS A REINFORCEMENT NEEDED?
00904  08CE  27 1D         BEQ   NOPE
00905  08D0  D5 7C         BIT B RSPFLG         HAS A RESPONSE TAKEN PLACE?
00906  08D2  27 19         BEQ   NOPE             IF NO, BRA NOPE

00908  08D4  17            TBA                  CLEAR BIT IN RENFLG
00909  08D5  43            COM A
00910  08D6  94 52         AND A RENFLG
00911  08D8  97 52         STA A RENFLG

00913  08DA  86 14         LDA A #DIVISR        RESTORE FREQDV TO BE COUNTED DOWN TO 0
00914  08DC  A7 48         STA A FREQDV,X

00916  08DE  8D 6C         BSR   RCREIN         RECORD THE REINFORCEMENT

00918  08E0  D7 5A         STA B BITPOS         SAVE B (UNIT BIT POSITION)
00919  08E2  A6 00         LDA A VI,X           A <- UNIT'S VI
00920  08E4  E6 08         LDA B SD,X           B <- UNIT'S SD
```

```
00922   C8E6   BD  07FF          JSR   DEVIAT          A <- ADJUSTED SAMPLE FROM DEVIAT SR

00924   C8E9   D6  5A            LDA B BITPOS          RESTORE B W/ UNIT BIT POSITION
00925   C8EB   A7  40            STA A CLOCK,X         M(CLOCK,X) <- A

00927   C8ED   54          NOPE  LSR B                 ALL UNITS BEEN CHECKED?
00928   C8EE   26  DB            BNE   LOOPA             IF NO, DO ANOTHER UNIT


00930   C8FC   39          ALL   RTS                   RETURN TO MAIN PROGRAM

00932                     ****************************** END OF RENCHK SR **********************************
```

```
00934                    ********************************************************************************
00935                    *                                  MAXCHK SR                                   *
00936                    ********************************************************************************

00938                    *    THIS SUBROUTINE IS CALLED BY VISORP MAIN PROGRAM TO TURN ON LED'S ON AN
00939                    *  INTERFACE BOARD WHEN A CERTAIN CONDOTION ARISES. LEDFLG IS PASSED INTO THIS
00940                    *  SR FROM RCREIN SR. LEDFLG CONTAINS UNITS' BIT POSITIONS HIGH WHICH NEED TO
00941                    *  HAVE THEIR ASSOCIATED LED'S LIT ON THE INTERFACE. IF NO BITS ARE SET IN
00942                    *  LEDFLG (LEDFLG = 0), THEN IMMEDIATELY RETURN TO VISORP. OTHERWISE, AN IMAGE
00943                    *  OF THE LED DATA REGISTER (MAXOUT), STORED IN PRVMAX, IS ORED WITH LEDFLG AND
00944                    *  THE RESULT STORED IN BOTH PRVMAX & MAXOUT. WRITING THIS INTO MAXOUT WILL
00945                    *  CAUSE THE LED'S TO LIGHT WHERE THERE IS A 1 IN THAT BIT POSITION AND A
00946                    *  BEEPER ON THE INTERFACE BOARD WILL ACTIVATE. THIS TELLS THE USER THAT THE
00947                    *  ANIMAL NEEDS TO BE TAKEN OUT OF THE CHAMBER. LEDFLG IS CLEARED AND SR RETURNS
00948                    *  TO VISORP.

00950    08F1    96  51  MAXCHK LDA A LEDFLG          ANY UNIT LEDS NEED LIGHTING?
00951    08F3    27  0A        BEQ   NO               IF NO, RETURN TO MAIN PROGRAM

00953    08F5    9A  78        ORA A PRVMAX          PREVIOUSLY LIT LED'S ARE KEPT ON BY ORING LEDFLG
00954    08F7    97  7B        STA A PRVMAX            AND PRVMAX TOGETHER AND SENDING RESULT TO PIA
00955    08F9    B7  8012      STA A MAXOUT
00956    08FC    7F  0051      CLR   LEDFLG

00958    08FF    39        NO  RTS                   RETURN TO MAIN PROGRAM

00960                    ****************************** END OF MAXCHK SR **********************************
```

```
00962                    ********************************************************************************
00963                    *                                BIGBEN SR                                     *
00964                    ********************************************************************************

00966                    *    THIS SUBROUTINE IS CALLED BY VISORP MAIN PROGRAM TO MAINTAIN ALL VARIABLES
00967                    *  USED AS "CLOCKS": FREQDV TABLE, CLOCK TABLE, SFRQDV, MINUTE, AND STIME.
00968                    *    THE FOLLOWING IS PERFORMED FOR ALL UNITS SERVICED IN THE SESSION. IF THE
00969                    *  UNIT'S ELEMENT IN VI TABLE IS ZERO, THEN THE ANIMAL IS TO RECEIVE CONTINUOUS
00970                    *  REINFORCEMENT DURING THE SESSION WHICH DOES NOT REQUIRE THE USE OF "CLOCKS",
00971                    *  THEREFORE CHECK THE NEXT UNIT'S FLAGS; OTHERWISE, CONTINUE. IF THE UNIT'S
00972                    *  BIT POSITION IN RENFLG IS SET, THE ANIMAL HAS STILL NOT RECEIVED ITS LAST
00973                    *  AVAILABLE REINFORCEMENT SO A NEW TIME INTERVAL CANNOT BE STARTED, THEREFORE
00974                    *  CHECK THE NEXT UNIT'S FLAGS; OTHERWISE, CONTINUE. IF THE UNIT'S BIT
00975                    *  POSITION IN RMXFLG IS SET, THE ANIMAL HAS BEEN ISSUED ALL OF ITS ALLOTTED
00976                    *  REINFORCEMENTS SO DISCONTINUE SERVICING THAT UNIT AND CHECK THE NEXT UNIT'S
00977                    *  FLAGS; OTHERWISE DECREMENT THE UNIT'S ELEMENT IN FREQDV. IF THE UNIT'S
00978                    *  ELEMENT IN FREQDV IS NOT ZERO, CHECK THE NEXT UNIT'S FLAGS; OTHERWISE RESET
00979                    *  IT TO DIVISR (20). FREQDV IS USED TO DIVIDE THE INTERRUPT FREQUENCY (20 HZ.)
00980                    *  DOWN TO 1 HZ. BECAUSE THE VARIABLE TIME INTERVALS ARE QUANTIZED TO THE
00981                    *  SECOND. DECREMENT THE UNIT'S ELEMENT IN CLOCK. IF IT IS NOT ZERO, CHECK THE
00982                    *  NEXT UNIT'S FLAGS; OTHERWISE, THE TIME INTERVAL REPRESENTED IN CLOCK HAS
00983                    *  ELLAPSED AND A REINFORCEMENT IS AVAILABLE AT THE NEXT RESPONSE FROM THAT
00984                    *  ANIMAL. TO INDICATE AN AVAILABLE REINFORCEMENT, SET THE UNIT'S BIT POSITION
00985                    *  IN RENFLG.
00986                    *    AFTER ALL THE UNITS' "CLOCKS" HAVE BEEN SERVICED, DECREMENT SFRQDV (THE
00987                    *  SESSION TIME'S FREQDV). IF SFRQDV IS NOT ZERO, RETURN TO VISORP; OTHERWISE,
00988                    *  RESET SFRQDV TO DIVISR (20) AND DECREMENT MINUTE. MINUTE IS USED TO DIVIDE
00989                    *  ITS EXECUTION FREQUENCY OF 1 HZ DOWN TO ONCE PER MINUTE BECAUSE STIME
00990                    *  (SESSION TIME) IS IN MINUTES. IF MINUTE IS NOT ZERO, RETURN TO VISORP;
00991                    *  OTHERWISE, RESET MINUTE TO 60 AND DECREMENT STIME. IF STIME IS NOT ZERO,
00992                    *  THE SESSION IS NOT OVER SO RETURN TO THE MAIN PROGRAM; OTHERWISE, TURN OFF
00993                    *  ALL OF THE REINFORCEMENT DEVICES, SET INTERRUPT BIT TO MASK INTERRUPT
00994                    *  REQUESTS AND TRANSFER CONTROL TO OUTPUT SR. THE EARLY TERMINATION ENTRY POINT
00995                    *  IS IN BIGBEN.

00997  0900  D6  68   BIGBEN LDA B UNITSP          B <- # OF UNITS BIT POSITION
00998  0902  DE  66          LDX   UNITS-1         X <- # OF UNITS(OFFSET FOR TABLES)

01000  0904  09       CLOCKS DEX                   X <- X - 1

01002  0905  A6  00          LDA A VI,X            IF VI = 0 THEN PROCESS NEXT UNIT
01003  0907  27  19          BEQ   NXTCLK                    ELSE CONTINUE

01005  0909  D5  52          BIT B RENFLG          IF RENFLG = 1, PROCESS NEXT UNIT
01006  090B  26  15          BNE   NXTCLK

01008  090D  D5  50          BIT B RMXFLG          IF RMXFLG = 1 THEN PROCESS NEXT UNIT
01009  090F  26  11          BNE   NXTCLK                      ELSE CONTINUE

01011  0911  6A  48          DEC   FREQDV,X        IF FREQDV = 0, THEN RESET
01012  0913  26  0D          BNE   NXTCLK            OTHERWISE, PROCESS NEXT UNIT
01013  0915  86  14          LDA A #DIVISR         RESTORE 'FREQDV' TO BE COUNTED DOWN
01014  0917  A7  48          STA A FREQDV,X          TO 0.

01016  0919  6A  40          DEC   CLOCK,X         UPDATE UNIT'S CLOCK
```

```
01017  0918  26 05          BNE   NXTCLK        IF CLOCK -= 0 THEN PROCESS NEXT UNIT

01019  0910  17              TBA                 IF CLOCK = 0 THEN SET BIT IN RENFLG
01020  091E  9A 52           ORA A RENFLG        (MEANS A REINFORCEMENT IS AVAILABLE AND WILL BE
01021  0920  97 52           STA A RENFLG        GIVEN AT NEXT RESPONSE)

01023  0922  54        NXTCLK LSR B               B <- SHIFT RIGHT(B)
01024  0923  26 DF          BNE   CLOCKS        IF ALL NOT PROCESSED, LOOP BACK FOR ANOTHER

01026  0925  7A 0056        DEC   SFRQDV        IF SESSION'S FREQDV = 0, THEN RESET
01027  0928  26 12          BNE   RETURN          ELSE, RETURN TO MAIN PROGRAM
01028  092A  86 14          LDA A #DIVISR       RESTORE SFRQDV TO BE COUNTED DOWN TO 0
01029  092C  97 56          STA A SFRQDV

01031  092E  7A 0058        DEC   MINUTE        DIVIDE 1HZ. BY 60 TO OBTAIN 1 MIN. TO
01032  0931  26 09          BNE   RETURN          TEST SESSION TIME.

01034  0933  86 3C          LDA A #60           RESTORE 'MINUTE' TO BE COUNTED DOWN TO 0
01035  0935  97 58          STA A MINUTE

01037  0937  7A 0057        DEC   STIME         UPDATE SESSION TIMER
01038  093A  27 01          BEQ   DONE          IF SESSION OVER, OUTPUT ACCUMULATED DATA
01039  093C  39        RETURN RTS                 IF NOT, RETURN TO MAIN PROGRAM

01041  093D  0C        DONE   CLC                NORMAL TERMINATION INDICATED BY C = 0
01042  093E  86 FF          LDA A #$FF
01043  0940  B7 801A        STA A OUTRNF        CLEAR SOLENOIDS
01044  0943  31              INS
01045  0944  31              INS
01046  0945  20 01          BRA   EARLY+1

01048  0947  0D        EARLY  SEC                EARLY TERMINATION ENTRY POINT (C = 1)
01049  0948  01              NOP
01050  0949  0F              SEI                 MASK INTERRUPT

01052  094A  20 2E          BRA   OUTPUT        PRINT ACCUMULATED DATA


01054                 ********** END OF BIGBEN SR **********
```

```
01056                  ********************************************************************************
01057                  *                                  RCREIN SR                                   *
01058                  ********************************************************************************

01060                  *    THIS SUBROUTINE IS CALLED FROM RSPCHK SR AND RENCHK SR TO RECORD THE
01061                  *  ISSUING OF A REINFORCEMENT. REGISTERS B AND X ARE PASSED INTO THIS SR TO
01062                  *  INDICATE TO WHICH UNIT THE REINFORCEMENT IS TO BE ASSOCIATED AND RECORDED.
01063                  *  REGISTER B, WHICH IS RETURNED UNCHANGED, CONTAINS A HIGH BIT AT ONE POSITION
01064                  *  SO THAT, FOR INSTANCE, TO RECORD A REINFORCEMENT FOR UNIT THREE, ITS
01065                  *  CONTENTS WOULD BE 0000 0100. REGISTER X, WHICH IS ALSO RETURNED UNCHANGED,
01066                  *  CONTAINS THE UNIT NUMBER MINUS ONE, SO THAT IN THE ABOVE EXAMPLE, IT WOULD
01067                  *  CONTAIN THE INTEGER TWO.
01068                  *    THE BIT SET IN B IS SET IN TRIGS. TRIGS IS PASSED TO IO SR WHERE IT CAUSES
01069                  *  THE ACTUAL TRIGGERING OF A REINFORCEMENT. X IS USED TO OBTAIN THE OFFSET IN
01070                  *  THE TWO TABLES: REINFS & REINMX. THE UNIT'S ELEMENT IN REINFS TABLE IS
01071                  *  INCREMENTED BY ONE. THE NEW COUNT IN THIS ELEMENT IS COMPARED TO THE UNIT'S
01072                  *  ELEMENT IN REINMX TABLE. IF THEY ARE NOT EQUAL, THIS SR RETURNS. IF THEY
01073                  *  ARE EQUAL, THIS MEANS THAT THE ANIMAL HAS RECEIVED ALL OF ITS ALLOTTED
01074                  *  REINFORCEMENTS. THE APPROPRIATE BIT IS SET IN LEDFLG WHICH IS PASSED TO
01075                  *  MAXCHK TO TURN ON AN LED TO ALERT THE USER OF THIS CONDITION. THE APPROPRIATE
01076                  *  BIT IN RMXFLG IS ALSO SET TO BE PASSED INTO RSPCHK SR & BIGBEN SR SO THAT
01077                  *  THAT UNIT WILL NO LONGER BE MONITORED OR SERVICED. THIS SR THEN RETURNS.

01079  094C  17        RCREIN TBA                    SET TRIGGER BIT
01080  094D  9A  6B           ORA A TRIGS
01081  094F  97  6B           STA A TRIGS

01083  0951  DF  5F           STX   SAVEX           SAVEX <- X
01084  0953  DF  5D           STX   OFFSET          CALC. OFFSET W/IN REINFS TABLE
01085  0955  78  005E         ASL   OFFSET+1           X <- X * 2
01086  0958  DE  5D           LDX   OFFSET

01088  095A  A6  31           LDA A REINFS+1,X
01089  095C  8B  01           ADD A #1              INCREMENT # OF REINFORCEMENTS
01090  095E  A7  31           STA A REINFS+1,X
01091  0960  24  02           BCC   SKIP11
01092  0962  6C  30           INC   REINFS,X

01094  0964  A1  11  SKIP11   CMP A REINMX+1,X      NO. OF REINFS.=MAX. FOR THAT UNIT?
01095  0966  26  0F           BNE   RCRDED
01096  0968  A6  30           LDA A REINFS,X
01097  096A  A1  10           CMP A REINMX,X
01098  096C  26  09           BNE   RCRDED

01100  096E  17                TBA                     IF YES, SET RMXFLG & LEDFLG BITS
01101  096F  9A  51           ORA A LEDFLG
01102  0971  97  51           STA A LEDFLG

01104  0973  9A  50           ORA A RMXFLG
01105  0975  97  50           STA A RMXFLG

01107  0977  DE  5F  RCRDED   LDX   SAVEX           X <- SAVEX

01109  0979  39              RTS                   RETURN
```

01111 ********** END OF RCREIN SR **********

```
01113                    ********************************************************************************
01114                    *                                OUTPUT SR                                     *
01115                    ********************************************************************************

01117                    *    THIS CODE SEGMENT IS EXECUTED FROM BIGBEN SR AT THE END OF A SESSION TO
01118                    *  PRINT, AT THE TERMINAL, THE RESULTS OF THE CURRENT SESSION. 'FINISH' IS
01119                    *  PRINTED IF THE SESSION TERMINATED NORMALLY (STIME MADE IT TO ZERO) AND
01120                    *  'EARLY FINISH' IS PRINTED IF THE SESSION WAS PREMATURELY TERMINATED BY
01121                    *  PRESSING THE RESET BUTTON ON THE COMPUTER AND THE PROGRAM RESTARTED AT $E000.
01122                    *  A HEADING IS PRINTED, FOLLOWED BY THE UNIT NUMBER, NUMBER OF RESPONSES,
01123                    *  AND NUMBER OF REINFORCEMENTS FOR EACH UNIT SERVICED DURING THE SESSION
01124                    *  ACCORDING TO THE CONTENTS OF UNITS. AFTER ALL SESSION DATA HAS BEEN PRINTED,
01125                    *  CONTROL IS TRANSFERED TO ASK.

01127  097A  24 05   CUTPUT BCC   NOTEAR          IF NORM. TERM., BRA NOTEAR
01128  097C  CE 0520         LDX   #MSG8          IF EARLY TERM., PRINT 'EARLY '
01129  097F  8D 5D           BSR   PDATA2

01131  0981  CE 04E8 NOTEAR LDX   #MSG3
01132  0984  8D 58           BSR   PDATA2         PRINT 'FINISH', CR, LF, BEEP
01133  0986  C6 0A           LDA B #10
01134  0988  8D 57           BSR   MYOUTS         SKIP 10 SPACES
01135  098A  CE 04F2         LDX   #MSG4
01136  098D  8D 4F           BSR   PDATA2         PRINT 'CHAMBER'
01137  098F  C6 0A           LDA B #10
01138  0991  8D 4E           BSR   MYOUTS         SKIP 10 SPACES
01139  0993  CE 04FA         LDX   #MSG5
01140  0996  8D 46           BSR   PDATA2         PRINT 'RESPONSES'
01141  0998  C6 0B           LDA B #11
01142  099A  8D 45           BSR   MYOUTS         SKIP 11 SPACES
01143  099C  CE 0504         LDX   #MSG6
01144  099F  8D 3D           BSR   PDATA2         PRINT 'REINFORCEMENTS', CR, LF

01146  09A1  CE 0000         LDX   #0
01147  09A4  DF 58           STX   COUNT
01148  09A6  DF 5F           STX   SAVEX

01150  09A8  DE 5F   DATOUT LDX   SAVEX
01151  09AA  7C 005C         INC   COUNT+1

01153  09AD  C6 0D           LDA B #13            SKIP 13 SPACES
01154  09AF  8D 30           BSR   MYOUTS
01155  09B1  86 30           LDA A #$30           PRINT UNIT NUMBERS
01156  09B3  9B 5C           ADD A COUNT+1
01157  09B5  BD E1D1         JSR   OUTEEE

01159  09B8  C6 0F           LDA B #15            SKIP 15 SPACES
01160  09BA  8D 25           BSR   MYOUTS
01161  09BC  EE 20           LDX   RESPS,X        GET DATA
01162  09BE  8D 28           BSR   PRINT          PRINT NUMBER OF RESPONSES

01164  09C0  50              NEG B                DIGIT COUNT IN B RETURNED BY PRINT SR
01165  09C1  CB 17           ADD B #23            SKIP TO COLUMN 53
01166  09C3  8D 1C           BSR   MYOUTS
01167  09C5  DE 5F           LDX   SAVEX
```

```
01168  09C7  EE 30        LDX   REINFS,X      GET DATA
01169  09C9  8D 1D        BSR   PRINT         PRINT NUMBER OF REINFORCEMENTS

01171  09CB  DE 5F        LDX   SAVEX
01172  09CD  08           INX                 INCREMENT ADDRESS VALUE TO NEXT UNIT'S DATA
01173  09CE  08           INX
01174  09CF  DF 5F        STX   SAVEX

01176  09D1  CE 052A      LDX   #CRLF         SKIP TO BEGINNING OF NEXT LINE
01177  09D4  8D 08        BSR   PDATA2

01179  09D6  D6 5C        LDA B COUNT+1       HAS ALL DATA BEEN PRINTED?
01180  09D8  D1 67        CMP B UNITS           IF YES, BRA ASK
01181  09DA  26 CC        BNE   DATOUT          IF NO, PRINT NEXT UNIT'S DATA
01182  09DC  20 6F        BRA   ASK

01184  09DE  7E E07E PDATA2 JMP   PDATA1        INTERMEDIATE JUMP TO PDATA1

01186                ****************************** END OF OUTPUT SR ********************************
```

```
01188                    ********************************************************************************
01189                    *                                MYOUTS SR                                     *
01190                    ********************************************************************************

01192                    *    THIS SUBROUTINE IS CALLED BY OUTPUT SR TO PRINT A SERIES OF SPACES. THE
01193                    *  OUTS SR IN THE MONITOR IS CALLED THE SAME NUMBER OF TIMES AS THE UNSIGNED
01194                    *  INTEGER IN B REGISTER. B IS ZERO UPON RETURN.

01196   09E1   BD  E0CC  MYOUTS JSR   OUTS           SKIP SPACES ACCORDING TO B REGISTER
01197   09E4   5A              DEC B
01198   09E5   26  FA          BNE   MYOUTS
01199   09E7   39              RTS

01201                    ****************************** END OF MYOUTS SR ***********************************
```

```
01203                    ********************************************************************************
01204                    *                                PRINT SR                                      *
01205                    ********************************************************************************

01207                    *    THIS SUBROUTINE IS CALLED BY OUTPUT SR TO PRINT, AT THE TERMINAL, THE
01208                    *  CONTENTS OF THE X REGISTER IN UNSIGNED DECIMAL INTEGER FORMAT. THE RANGE
01209                    *  OF NUMBERS IS 0 -> 65535 WITH THE FIRST CHARACTER POSITIONED AT THE CURSOR.
01210                    *  LEADING ZEROS ARE SUPPRESSED BEFORE NON-ZERO DIGITS AND A SINGLE ZERO IS
01211                    *  PRINTED FOR X = 0. THE 48-BYTE DATA STRUCTURE: TABLE IS REQUIRED FOR
01212                    *  EXECUTION OF THIS SR. THIS SR IS A MODIFICATION OF A SR WRITTEN BY
01213                    *  FRANSISCO BASCUNAN.


01215                    *
01216                    *    THIS PART CONVERTS THE BINARY NUMBER INTO DECIMAL
01217                    *
01218  09E8  DF  61      PRINT    STX  NUMBER     SAVE THE DATA.
01219  09EA  CE  04A0             LDX  #TABLE     GET ADDRESS OF DECIMAL CONVERTION TABLE.
01220  09ED  4F                   CLRA            CLEAR THE STORAGE FOR THE RESULT OF CONVERSION.
01221  09EE  97  63               STAA NUM
01222  09F0  97  64               STAA NUM+1
01223  09F2  97  65               STAA NUM+2
01224  09F4  D6  61               LDAB NUMBER     GET MSB.
01225  09F6  96  62               LDAA NUMBER+1   GET LSB.

01227  09F8  54          L12      LSRB            ROTATE BOTH REGISTERS TO TEST EACH BIT IN CARRY.
01228  09F9  46                   RORA
01229  09FA  24  19               BCC  NOTADD     IF CARRY = 0, DO NOT ADD.
01230  09FC  37                   PSHB            SAVE THE NUMBERS.
01231  09FD  36                   PSHA
01232  09FE  A6  02               LDAA 2,X        GET LSD FROM TABLE.
01233  0A00  98  65               ADDA NUM+2
01234  0A02  19                   DAA
01235  0A03  97  65               STAA NUM+2      SAVE THE TWO LSD AFTER THE DAA.
01236  0A05  A6  01               LDAA 1,X        GET NEXT TWO DIGITS.
01237  0A07  99  64               ADCA NUM+1
01238  0A09  19                   DAA
01239  0A0A  97  64               STAA NUM+1
01240  0A0C  A6  00               LDAA 0,X        GET THE LAST TWO NUMBERS.
01241  0A0E  99  63               ADCA NUM
01242  0A10  19                   DAA
01243  0A11  97  63               STAA NUM
01244  0A13  32                   PULA
01245  0A14  33                   PULB            RESTORE THE NUMBER.
01246  0A15  08          NOTADD   INX             POINT TO THE NEXT POSITION ON THE TABLE.
01247  0A16  08                   INX
01248  0A17  08                   INX
01249  0A18  4D                   TSTA
01250  0A19  26  DD               BNE  L12
01251  0A1B  5D                   TSTB            IF A AND B ARE BOTH = 0, NO MORE ADDING IS NEEDED.
01252  0A1C  26  DA               BNE  L12
01253                    *
01254                    *    THIS PART PRINTS THE DATA AFTER IT HAS BEEN TRANSLATED INTO DECIMAL.
01255                    *
01256  0A1E  5F                   CLRB            USED AS LEADING ZERO FLAG.
01257  0A1F  CE  0063             LDX  #NUM
```

```
01259  0A22  A6  00    L9      LDAA 0,X       GET LEFT PART OF BYTE.
01260  0A24  44              LSRA
01261  0A25  44              LSRA
01262  0A26  44              LSRA
01263  0A27  44              LSRA
01264  0A28  26  03          BNE  *+2+3      SEE IF ZERO.
01265  0A2A  5D              TSTB            SEE IF THIS ZERO IS LEADING.
01266  0A2B  27  02          BEQ  *+2+2      IF LEADING, DO NOT PRINT.
01267  0A2D  8D  17          BSR  PRINT1
01268  0A2F  A6  00          LDAA 0,X        GET RIGHT PART OF BYTE.
01269  0A31  84  0F          ANDA #$F        MASK MSN OF BYTE.
01270  0A33  26  03          BNE  *+2+3      SEE IF ZERO.
01271  0A35  5D              TSTB            SEE IF THIS ZERO IS LEADING.
01272  0A36  27  02          BEQ  *+2+2      IF LEADING, DO NOT PRINT.
01273  0A38  8D  0C          BSR  PRINT1
01274  0A3A  08              INX
01275  0A3B  8C  0066        CPX  #NUM+3     SEE IF ALL #S HAVE BEEN PRINTED.
01276  0A3E  26  E2          BNE  L9         GET NEXT BYTE OR RETURN.

01278  0A40  5D              TSTB            SEE IF ALL WERE LEADING ZEROS.
01279  0A41  26  02          BNE  *+2+2
01280  0A43  8D  01          BSR  PRINT1     IF SO, PRINT A ZERO.
01281  0A45  39              RTS
01282                 *
01283                 *
01284  0A46  8B  30  PRINT1  ADDA #'0        GET ASCII CODE.
01285  0A48  BD  E101        JSR  OUTEEE
01286  0A4B  5C              INCB            A NUMBER HAS BEEN PRINTED. NO MORE LEADING ZEROS.
01287  0A4C  39              RTS

01289                 ****************************** END OF PRINT SR ********************************
```

```
01291                    ********************************************************************************
01292                    *                                ASK SR                                        *
01293                    ********************************************************************************

01295                    *    THIS CODE SEGMENT IS EXECUTED AFTER OUTPUT SR. THE USER IS ASKED IF
01296                    *  ANOTHER SESSION IS TO BE SET UP AND RUN. AN AFFIRMATIVE RESPONSE ('Y')
01297                    *  CAUSES THIS TO HAPPEN WHILE A NEGATIVE RESPONSE ('N') RESULTS IN A CONTROL
01298                    *  TRANSFER TO THE COMPUTER SYSTEMS MONITOR. EITHER RESPONSE MAY BE PRECEDED
01299                    *  BY ONE OR MORE BLANKS. IF A RESPONSE IS NEITHER 'Y' OR 'N', THE QUESTION
01300                    *  ('ANOTHER?') IS REITERATED.

01302    0A4D    CE  0515  ASK     LDX   #MSG7          PRINT 'ANOTHER?'
01303    0A50    8D  8C            BSR   PDATA2
01305    0A52    BD  07B6          JSR   SINEEE         INPUT REPLY-WAIT FOR ASCII -= $20
01306    0A55    81  59            CMP A #'Y            ='Y'?
01307    0A57    27  08            BEQ   AGIN            IF YES, BRANCH AGAIN
01308    0A59    81  4E            CMP A #'N            IF NO, ='N'?
01309    0A5B    27  0A            BEQ   NOMORE           IF YES, BRANCH NOMORE
01310    0A5D    86  3F            LDA A #'?              IF NO, PRINT '?'  AND GUESS AGAIN
01311    0A5F    BD  E1D1          JSR   OUTEEE
01312    0A62    20  E9            BRA   ASK
01314    0A64    7E  0602  AGIN    JMP   SESHUN         JUMP TO BEGINNING OF VISORP
01316    0A67    7E  E0E3  NOMORE  JMP   MONITR         JUMP TO MONITOR

01318                    ********************************** END OF ASK SR **********************************
```

```
01320                   ********************************************************************************
01321                   *                                  INTRPT SR                                   *
01322                   ********************************************************************************

01324                   *    THIS CODE SEGMENT IS EXECUTED IMMEDIATELY AFTER EVERY INTERRUPT REQUEST.
01325                   *  WAIFLG IS ALWAYS ZERO PRIOR TO INTRPT'S EXECUTION SO THAT COMPLEMENTING
01326                   *  SETS IT TO $FF. THE READ FROM MAXOUT RESETS THE INTERRUPT REQUEST CONTROL
01327                   *  LINE ON THE SYSTEM BUSS TO HIGH.

01329   0A6A    73  0055  INTRPT COM   WAIFLG          TOGGLE WAIFLG TO $FF (SHOWS IRQ OCCURED)
01330   0A6D    B6  8012        LDA A MAXOUT          RESET IRQ BY READING MAXOUT

01332   0A7C    3B              RTI                   RETURN FROM INTERRUPT

01334                   ***************************** END OF INTRPT SR *********************************
```

```
01336    0E00                        ORG    $0E00
01337    0E00    7E  0947            JMP    EARLY          EASIER TO REMEMBER ADDRESS
01338                                END
```

TOTAL ERRORS 0

LISTING OF OBJECT FILE

```
04A0  00 00 01 00 00 02 00 00 04 00 00 08 00 00 16 00 *
04B0  00 32 00 00 64 00 01 28 00 02 56 00 05 12 00 10 *
04C0  24 C0 20 48 00 40 96 0C 81 92 01 63 84 03 27 68 *
04D0  0D 0A 0A 0A 0A 0A 56 49 53 4F 52 50 0D 0A 0A 04 *
04E0  54 41 52 54 0D 0A 0A 04 46 49 4E 49 53 48 0D 0A *
04F0  07 04 43 48 41 4D 42 45 52 04 52 45 53 50 4F 4E *
0500  53 45 53 04 52 45 49 4E 46 4F 52 43 45 40 45 4E *
0510  54 53 CD 0A 04 0D 0A 41 4E 4F 54 48 45 52 3F 04 *
0520  0A 0D 45 41 52 4C 59 20 04 3F 0D 0A 04 *
06C0  01 CF 8D 1D 8D 35 8D 64 0E 96 55 27 FC 7F 00 55 *
0610  BD C8 5B BD 08 8A BD 08 BF BD 08 F1 BD 09 00 20 *
0620  E8 CE FF AD FF 80 12 7F 80 12 86 FF C6 04 F7 80 *
0630  19 B7 80 1A F7 80 1B B7 80 1A 39 CE 00 00 6F 00 *
0640  08 8C 00 A0 26 F8 86 04 97 67 CE 03 FF DF 69 CE *
0650  52 0D DF 76 CE CB B9 DF 78 CE 0A 6A FF A0 00 86 *
0660  1E 57 57 86 14 97 56 86 3C 97 58 39 CE 04 00 8D *
067C  E0 7E 8D 07 B6 81 23 26 06 7E 07 56 BD 07 B6 81 *
0680  53 26 03 7E 07 CB 81 54 26 03 7E 07 70 81 5E 26 *
0690  05 31 31 7E 06 04 81 58 26 03 7E E0 E3 81 31 2C *
06A0  02 20 45 80 30 91 67 2E F8 97 5E 7A 00 5E 5F 0D *
06B0  59 4A 26 FC 53 D4 53 D7 53 BD E1 AC 81 20 26 28 *
06C0  C6 04 BD 07 94 E6 6E A6 6D DF 5F DE 5D E7 00 BD *
06D0  07 C5 AB 00 A7 00 DE 5F 5F A6 6C 27 0E C6 64 81 *
06E0  01 27 08 58 81 02 27 03 7E 07 C0 DE 5D EB 00 25 *
06F0  F7 E7 00 C6 03 BD 07 94 E6 6E A6 6D DE 5D E7 08 *
0700  BD 07 C5 AB 08 81 55 22 DF A7 08 78 00 5E C6 04 *
0710  BD 07 94 E6 6E A6 6D DF 5F DE 5D E7 11 BD 07 C5 *
0720  AB 11 A7 11 DE 5F A6 6C DE 5D BD 07 C5 5F A7 10 *
0730  48 48 59 AB 10 48 59 AB 11 A7 11 C9 00 E7 10 96 *
0740  5E 44 4C 5F 0D 59 4A 26 FC DA 53 D7 53 CE 05 2A *
0750  BD E0 7E 7E 06 7C BD 07 86 81 31 2D 04 81 38 2F *
0760  09 CE 05 29 BD E0 7E 7E 06 72 80 30 97 67 20 DD *
0770  C6 C4 8D 20 A6 6D 8D 4D AB 6E 97 74 5F A6 6C 27 *
0780  08 C6 64 81 01 27 05 58 81 02 26 34 DB 74 23 30 *
0790  D7 57 20 B9 4F 97 6C 97 6D CE FF FF 8D 18 81 30 *
07A0  2D 1C 81 39 2E 18 80 30 08 5A 27 12 A7 6E BD E1 *
07B0  AC 81 20 26 E9 39 8D E1 AC 81 20 27 F9 39 31 31 *
07C0  CE 05 29 20 8B 16 48 48 1B 48 39 D6 53 96 67 4C *
07D0  4A 54 25 FC 4D 26 E9 96 67 5F 0D 59 4A 26 FC D7 *
07E0  68 CE 04 E0 BD E0 7E 8D 01 39 CE 00 00 A6 00 E6 *
07F0  08 8D 0C A7 40 86 14 A7 48 08 9C 66 26 EF 39 97 *
0800  7A DF 5F DE 69 A6 A0 05 8C FF FF 26 03 CE 03 FF *
0810  DF 69 DE 5F 97 71 D7 72 86 F8 97 73 4F 5F 58 49 *
0820  78 C0 72 24 04 D8 71 89 00 7C 00 73 26 F0 97 74 *
0830  D7 75 58 49 D8 75 99 74 78 00 79 79 00 78 79 00 *
0840  77 79 00 76 24 0E C6 01 DA 79 D7 79 16 96 7A 10 *
0850  24 08 4F 39 9B 7A 24 02 86 FF 39 B6 80 18 43 26 *
0860  04 57 59 20 0A 16 73 00 59 94 59 D7 59 97 54 86 *
0870  FF B7 80 1A 96 6B 27 11 86 01 95 6B 26 03 48 20 *
0880  F9 43 B7 80 1A 94 6B 97 6B 39 96 54 97 7C 27 2E *
0890  D6 68 DE 66 09 D5 54 27 20 D5 50 26 1C A6 00 26 *
08A0  03 BD 09 4C DF 5F DF 5D 78 00 5E DE 5D A6 21 8B *
08B0  01 A7 21 24 02 6C 20 DE 5F 54 26 D8 D7 54 39 96 *
08C0  52 27 2D 96 7C 27 29 D6 68 DE 66 09 D5 52 27 1D *
```

```
08D0  D5 7C 27 19 17 43 94 52 97 52 86 14 A7 48 8D 6C *
08E0  D7 5A A6 00 E6 08 BD 07 FF D6 5A A7 40 54 26 DB *
08F0  39 96 51 27 0A 9A 7B 97 7B B7 80 12 7F 00 51 39 *
0900  D6 68 DE 66 09 A6 00 27 19 D5 52 26 15 D5 50 26 *
0910  11 6A 48 26 0D 86 14 A7 48 6A 40 26 05 17 9A 52 *
0920  97 52 54 26 DF 7A 00 56 26 12 86 14 97 56 7A 00 *
0930  58 26 09 86 3C 97 58 7A 00 57 27 01 39 0C 86 FF *
0940  B7 80 1A 31 31 20 01 0C 01 0F 20 2E 17 9A 6B 97 *
0950  6B DF 5F DF 5D 78 00 5E DE 5D A6 31 8B 01 A7 31 *
0960  24 02 6C 30 A1 11 26 0F A6 30 A1 10 26 09 17 9A *
0970  51 97 51 9A 50 97 50 DE 5F 39 24 05 CE 05 20 8D *
0980  5D CE 04 E8 8D 58 C6 0A 8D 57 CE 04 F2 8D 4F C6 *
0990  0A 8D 4E CE 04 FA 8D 46 C6 0B 8D 45 CE 05 04 8D *
09A0  3D CE C0 00 DF 5B DF 5F DE 5F 7C 00 5C C6 0D 8D *
09B0  30 86 30 9B 5C BD E1 D1 C6 0F 8D 25 EE 20 8D 28 *
09C0  50 CB 17 8D 1C DE 5F EE 30 8D 1D DE 5F 08 08 DF *
09D0  5F CE 05 2A 8D 08 D6 5C D1 67 26 CC 20 6F 7E E0 *
09E0  7E 8D E0 CC 5A 26 FA 39 DF 61 CE 04 A0 4F 97 63 *
09F0  97 64 97 65 D6 61 96 62 54 46 24 19 37 36 A6 02 *
0A00  9B 65 19 97 65 A6 01 99 64 19 97 64 A6 00 99 63 *
0A10  19 97 63 32 33 08 08 08 4D 26 0D 5D 26 DA 5F CE *
0A20  00 63 A6 00 44 44 44 44 26 03 5D 27 02 8D 17 A6 *
0A30  00 84 0F 26 03 5D 27 02 8D 0C 08 8C 00 66 26 E2 *
0A40  5D 26 02 8D 01 39 8B 3C BD E1 D1 5C 39 CE 05 15 *
0A50  8D 8C 8D 07 B6 81 59 27 08 81 4E 27 0A 86 3F 8D *
0A60  E1 D1 20 E9 7E 06 02 7E E0 E3 73 00 55 B6 80 12 *
0A70  3B *
0E00  7E 09 47 *
```

*** SYMBOL TABLE ***

ADD	0854
ADJSTD	085A
AGIN	0A64
ALL	08F0
ALLTST	0053
ASK	0A4D
BIGBEN	0900
BIT	0881
BITPCS	005A
CLEAR	063B
CLOCK	0040
CLOCKS	0904
COMAGN	06E8
COUNT	0058
CRLF	052A
DATCUT	09A8
DEVIAT	07FF
DIVISR	0014
DONE	093D
DSPTCH	0609
EARLY	0947
ENTER	079E
EVRCLR	063E
EXIT	0889
FIRSTP	067F
FIRST2	520D
FREQDV	0048
HAVDAT	0865
HUH1	07C0
HUH2	07BE
INEEE	E1AC
INITAL	0621
INPUT	066C
INRSPS	8018
INTERM	006E
INTRPT	0A6A
INUNIT	067C
INVALD	0761
IO	085B
IOV	A000
K	0400
LEDFLG	0051
LOOP	087A
LOOPA	08CB
L12	09F8
L9	0A22
MAXCHK	08F1
MAXCUT	8012
MCND	0071
MINUTE	0058
MCNITR	E0E3
MOVE1	07EA
MPLR	0072
MPYCNT	0073

MPYNCR	0829
MPYRLP	081E
MSG1	04D0
MSG2	04E0
MSG3	04E8
MSG4	04F2
MSG5	04FA
MSG6	0504
MSG7	0515
MSG8	0520
MYOUTS	09E1
NEWLIN	074D
NEXT1	08B9
NO	08FF
NOLOGO	0794
NOMORE	0A67
NOPE	08ED
NOTADD	0A15
NOTEAR	0981
NOTRST	0810
NOTRU	08BE
NOTS	0686
NOTT	068D
NOTX	069D
NOT5E	0696
NUM	0063
NUMBER	0061
NXTCLK	0922
OFFSET	0050
OK	078C
OK1	06EB
OK2	06A3
OUTEEE	E1D1
OUTPUT	097A
OUTRNF	801A
OUTS	E0CC
PDATA1	E07E
PDATA2	09DE
PCINTR	0069
POUNDS	0756
PRINT	09E8
PRINT1	0A46
PRVMAX	0078
PRVRSP	0059
QUEST	0529
RCRDED	0977
RCREIN	094C
REINFS	0030
REINMX	0010
RENCHK	088F
RENFLG	0052
RESPS	0020
RETURN	093C
RMXFLG	0050
RNDSGN	0076
RSPCHK	088A

RSPFLG	007C
SAMPLE	00A0
SAVEX	005F
SAVVI	007A
SD	0008
SECND2	CBB9
SESHUN	0602
SFRQDV	0056
SHIFTB	07DB
SHIFTO	07D0
SHIFT1	0680
SHIFT2	0745
SINEEE	07B6
SKIPA	08B7
SKIPB	08A4
SKIP11	0964
START	07CB
STDFLT	001E
STIME	0057
SUPERC	076A
TABLE	04A0
TBLSIZ	0400
TEMPA	0074
TEMPB	0075
TIME10	07C5
TPARAM	0770
TRIGGR	086F
TRIGS	006B
TRURSP	0054
TRYAGN	0672
TSTBIT	0894
UDFLT	0004
UNITS	0067
UNITSP	0068
VI	0000
VISORP	0600
WAIFLG	0055

NORMHEX

program listing and output

prepared for

Dr. Daniel W. Scott III

CSCI 488

by

Daniel Paul Long

February 28, 1979

Note: Each run of NORMHEX produces a unique set of output (the static table called VALUES in NORMHEX). The last 3 pages of the following 22 pages contains the actual data base, in hexadecimal, used by the DEVIAT ALC subroutine and all testing programs which followed (NORMTEST and DISTRIB).

```
*PL/C SORMGIN=(2,80,1),TIME=(5,0)
*OPTIONS IN EFFECT*      PAGES=030,TIME=(005,000),ERRORS=(050,050),SORMGIN=(002,080,001),LINECNT=060,BOUNDARY,
*OPTIONS IN EFFECT*      FLAGW,NOXREF,NOATR,NOLIST,UDEF,SOURCE,DUMP,NODUMPARRAY,NOM91,NOCOMMENTS,CHECK

      /*    NORMHEX    */                                                    PL/C-R6.6000  02/28/79 21:20 PAGE    1
   STMT LEVEL NEST BLOCK      SOURCE STATEMENT                                                          ID FIELD

                              /*    NORMHEX    */
                              /*  GENERATE RANDOM NORMALLY DISTRIBUTED VALUES (UNSIGNED) AND CONVERT TO HEX*/
                              /*        WHERE FF IS 3 STD. DEV. FROM MEAN 0      */

        1                     SAMPLES:PROC OPTIONS(MAIN);
        2     1         1     DCL (MEAN,STD_DEV) FIXED(9,3), VALUES(1024) FIXED(6), ((I,J) FIXED,
                              (X,Y) FIXED(10,10)) BINARY, HEXVALUE CHAR(2) VARYING, HEXCONV(0:15) CHAR(1)
                              INIT('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'),
                              NORMAL ENTRY RETURNS(FIXED(6)),LINES(-255:255) BINARY INIT((511)0),
                              SUM_OF_THE_SQUARES FIXED(8) INIT(0);

                              /*  USE 'TIME' AS RANDOM SEED  */

        3     1         1     GET STRING(TIME) EDIT(X)(F(9,9));

                              /*  FILL ARRAY WITH NORMALLY DISTRIBUTED VALUES  */

        4     1         1     DO I=LBOUND(VALUES,1) TO HBOUND(VALUES,1);
        5     1    1    1       VALUES(I) = NORMAL;
        6     1    1    1       DO J=1 BY 1 WHILE (VALUES(I) > 255 | VALUES(I) < -255);
        7     1    2    1         VALUES(I) = NORMAL; END;
        9     1    1    1       SUM_OF_THE_SQUARES = SUM_OF_THE_SQUARES + VALUES(I)**2;
       10     1    1    1     END;

                              /*  PRINT DECIMAL VALUES  */

       11     1         1     PUT LIST('DECIMAL'); PUT SKIP(2);
       13     1         1     PUT LIST(VALUES);

                              /*  CALCULATE MEAN  */

       14     1         1     MEAN = SUM(VALUES) / DIM (VALUES,1);

                              /*  CALCULATE STANDARD DEVIATION  */

       15     1         1     STD_DEV=SQRT((SUM_OF_THE_SQUARES-DIM(VALUES,1)*MEAN**2)/(DIM(VALUES,1)-1));
```

```
STMT LEVEL NEST BLOCK      SOURCE STATEMENT                                    ID FIELD

                           /*  FILL ARRAY LINES  */
  16    1        1         DO I=LBOUND(VALUES,1) TO HBOUND(VALUES,1);
  17    1    1   1           DO J=-255 TO 255 WHILE (VALUES(I) ¬= J); END;
  19    1    1   1           LINES(J) = LINES(J) + 1;
  20    1    1   1           END;

  21    1        1         PUT PAGE LIST('DISTRIBUTION HISTOGRAPH'); PUT SKIP(2);
  23    1        1         ON ENDPAGE;

                           /*  GRAPH DISTR. HISTOGRAPH  */
  24    2        2         DO I=-255 TO 255;
  26    1    1   1           PUT SKIP EDIT(I,SUBSTR(REPEAT('*',131),1,LINES(I)*2))(F(4),X(1),A);
  27    1    1   1           END;

  28    1        1         ON ENDPAGE SYSTEM;
  29    1        1         PUT SKIP(15) DATA(LINES(0));
```

/* NORMHEX */

```
STMT LEVEL NEST BLOCK      SOURCE STATEMENT                                    ID FIELD
                           /*  MAKE VALUES UNSIGNED BEFORE CONVERSION TO HEX  */
     30    1        1      VALUES = ABS(VALUES);
     31    1        1      PUT PAGE LIST('CONVERSION TO HEXADECIMAL'); PUT SKIP(2);
     33    1        1      DO I=LBOUND(VALUES,1) TO HBOUND(VALUES,1);
                           /*  TRANSLATE 1ST HEX DIGIT  */
     34    1    1   1        DO J=240 TO 0 BY -16 WHILE (VALUES(I) < J); END;
     36    1    1   1          HEXVALUE = HEXCONV(J / 16 );
     37    1    1   1          VALUES(I) = VALUES(I) - J;
                           /*  TRANSLATE 2ND HEX DIGIT  */
     38    1    1   1        DO J=15 TO 0 BY -1 WHILE (VALUES(I) < J); END;
     40    1    1   1          HEXVALUE = HEXVALUE || HEXCONV(J);
                           /*  PRINT CONVERSION  */
     41    1    1   1        PUT LIST(HEXVALUE);
     42    1    1   1      END;
                           /*  SR THAT RETURNS SCALED NORM. DISTR. VALUE  */
     43    1        1      NORMAL:PROC RETURNS(FIXED(6));
IN   43                ERROR  SYOC RETURN ATTRIBUTES OVERRIDE DEFAULT
     44    2        3        Y=RAND(X); X=RAND(Y);
     46    2        3        IF X=0 THEN X=RAND(X); IF Y=0 THEN Y=RAND(Y);
     50    2        3        RETURN (ROUND(LOG(X) * SIN(6.28319 * Y) * 124,0));
     51    2        3      END NORMAL;
     52    1        1      END SAMPLES;

WARNING  CGOC NO FILE SPECIFIED. SYSIN/SYSPRINT ASSUMED.
WARNING  CG13 RAND BUILT-IN FUNCTION USED.
WARNING  CG13 RAND BUILT-IN FUNCTION USED.
WARNING  CG13 RAND BUILT-IN FUNCTION USED.
WARNING  CG13 RAND BUILT-IN FUNCTION USED.
```

DECIMAL

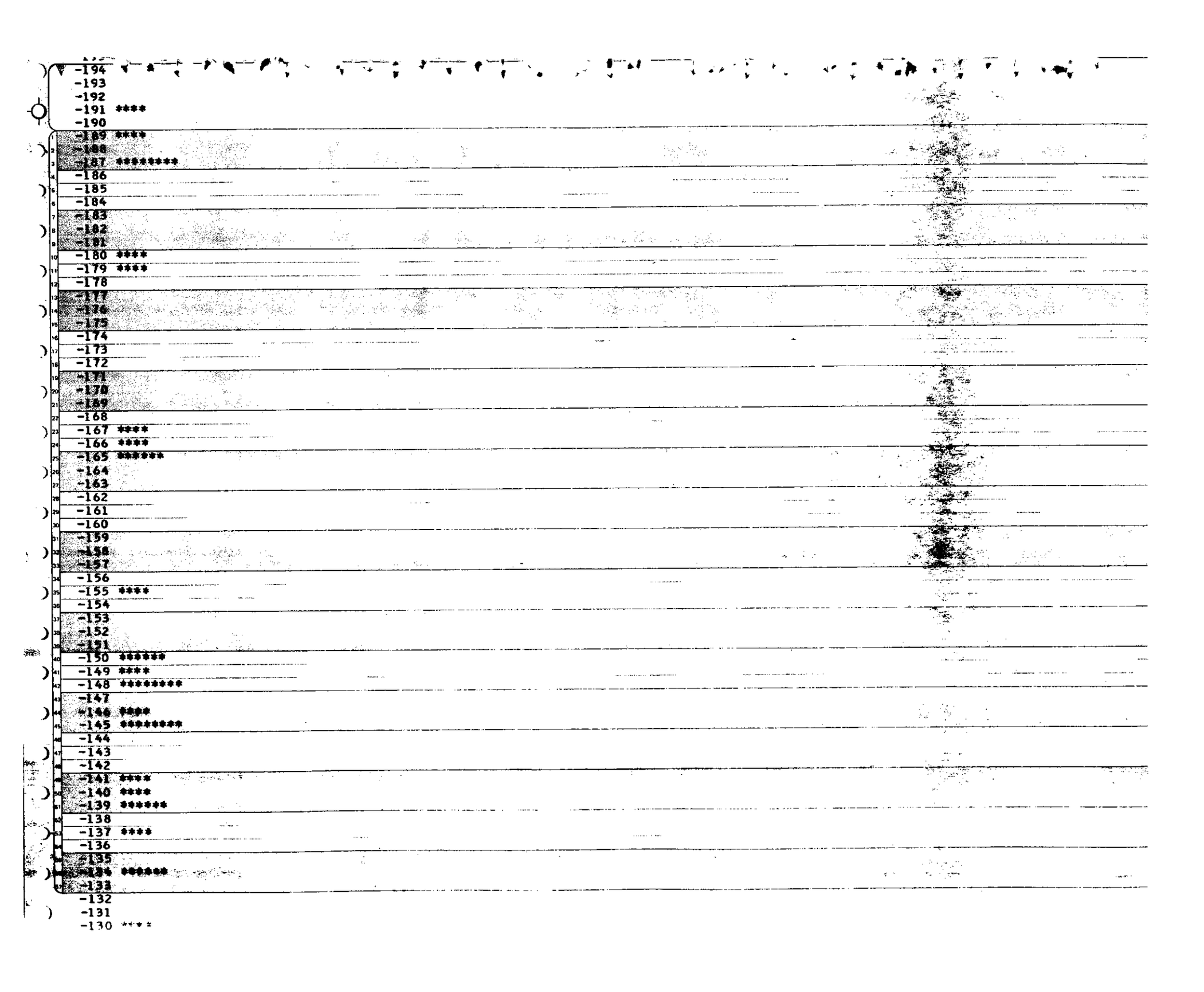
-215	1	34	36	2	10
21	2	-5	1	82	-113
-35	-11	-78	-150	31	-1
26	2	177	249	162	25
52	-37	-80	41	2	83
40	163	-75	41	-96	68
12	-77	1	16	-33	-16
18	112	128	-165	-67	136
9	-107	30	-57	30	-207
-139	-134	37	50	-12	-19
131	-198	12	32	-118	-180
-43	11	-145	91	8	64
223	33	-3	-64	13	-249
-8	-216	-28	237	-78	23
-33	49	24	23	-88	7
-38	99	-69	29	-42	21
47	-18	145	-8	58	-28
18	-101	-105	155	46	70
-67	193	-17	-111	-26	128
-7	63	64	-46	60	1
-187	74	-5	-38	-43	177
219	38	-51	8	-33	0
-5	-23	-97	41	-1	-54
121	122	116	20	-58	-89
-38	-99	15	-75	-4	13
66	0	-118	-12	4	18
-126	0	-11	62	-108	-228
-37	30	-19	44	40	-75
-12	-128	60	-145	0	135
6	-26	76	4	-83	18
159	88	128	23	16	100
-42	-148	-13	41	64	-9
-228	-107	-3	-10	-55	86
-127	-4	97	79	71	-24
77	87	0	48	-57	-15
-21	175	10	68	44	-8
0	117	-15	9	78	35
126	-28	8	-10	2	5
11	-2	-3	93	-24	52
52	-80	-26	-37	-71	-31
48	0	-167	-166	-35	-59
-48	0	-78	-45	43	0
-13	-66	18	169	110	3
59	31	-149	-57	66	-57
-141	-90	-39	68	-56	-36
-110	-10	168	0	-7	-13
-9	-32	-191	-53	5	-109
17	-16	-88	1	91	-82
-19	-24	37	-28	-1	21
-146	12	-137	46	31	22
-7	-244	92	125	-27	44
-71	-148	26	65	30	195
0	32	-3	-62	-56	-50
-4	21	13	-3	56	94
48	5	-53	16	-5	162
-5	-140	-102	6	-71	170
-20	8	-130	194	-4	-216
80	69	-29	-46	103	126

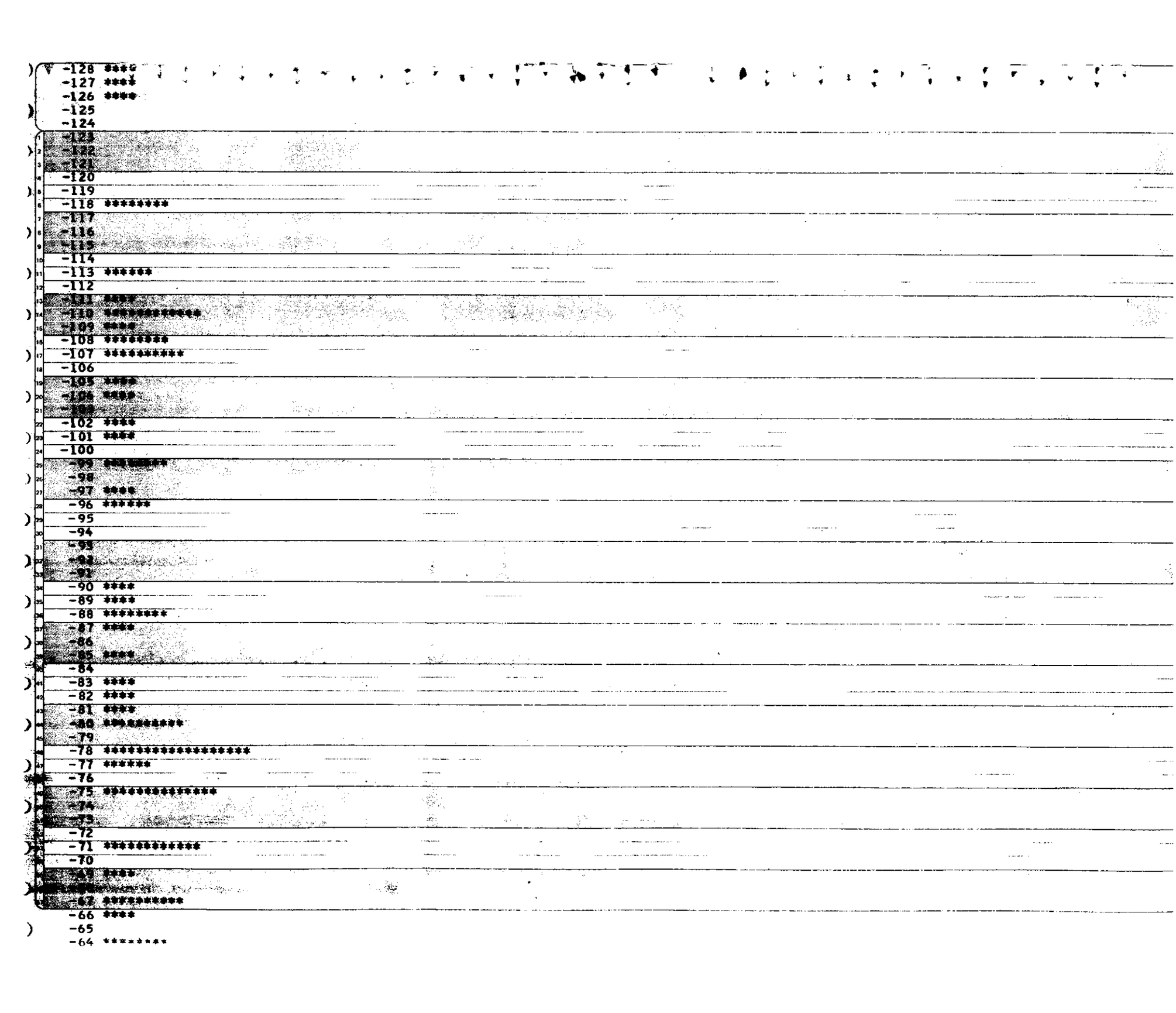
-87	7	10	-29	-206	-15
1	-43	122	24	145	-23
1	-3	-8	209	-8	-43
41	-13	95	178	8	-39
-49	-179	236	-1	151	-22
1	107	18	52	8	-25
-25	-23	-4	5	17	5
14	-110	58	-64	77	4
15	76	-24	-108	25	-187
59	-14	12	51	-57	174
-155	124	-213	-210	7	64
22	27	-48	-22	-78	2
-6	-18	-104	-36	-19	-22
-1	-58	-110	-12	7	29
88	-189	-3	45	51	20
24	1	246	-61	-3	18
0	-21	-12	2	0	-1
29	-211	12	112	95	-39
-85	-196	88	11	-14	95
-99	-15	-211	-24	6	-53
94	-81	-22	-48	15	28
-6	116	-5	-11	4	3
184	144	-215	1	34	36
2	10	21	2	-5	1
82	-113	-35	-11	-78	-150
31	-1	26	2	177	249
162	25	52	-37	-80	41
2	83	40	163	-75	41
-96	68	12	-77	1	16
-33	-16	18	112	128	-165
-67	136	9	-107	30	-57
30	-207	-139	-134	37	50
-12	-19	131	-198	12	32
-118	-180	-43	11	-145	91
8	64	223	33	-3	-64
13	-249	-8	-216	-28	237
-78	23	-33	49	24	23
-88	7	-38	99	-69	29
-42	21	47	-18	145	-8
58	-28	18	-101	-105	155
46	70	-67	193	-17	-111
-26	128	-7	63	64	-46
60	1	-187	74	-5	-38
-43	177	219	38	-51	8
-33	0	-5	-23	-97	41
-1	-54	121	122	116	20
-58	-89	-38	-99	15	-75
-4	13	66	0	-118	-12
4	18	-126	0	-11	62
-108	-228	-37	30	-19	44
40	-75	-12	-128	60	-145
0	135	6	-26	76	4
-83	18	159	88	128	23
16	100	-42	-148	-13	41
64	-9	-228	-107	-3	-10
-55	86	-127	-4	97	79
71	-24	77	87	0	48
-57	-15	-21	175	10	68
44	-8	0	117	-15	9
78	35	126	-28	8	-10

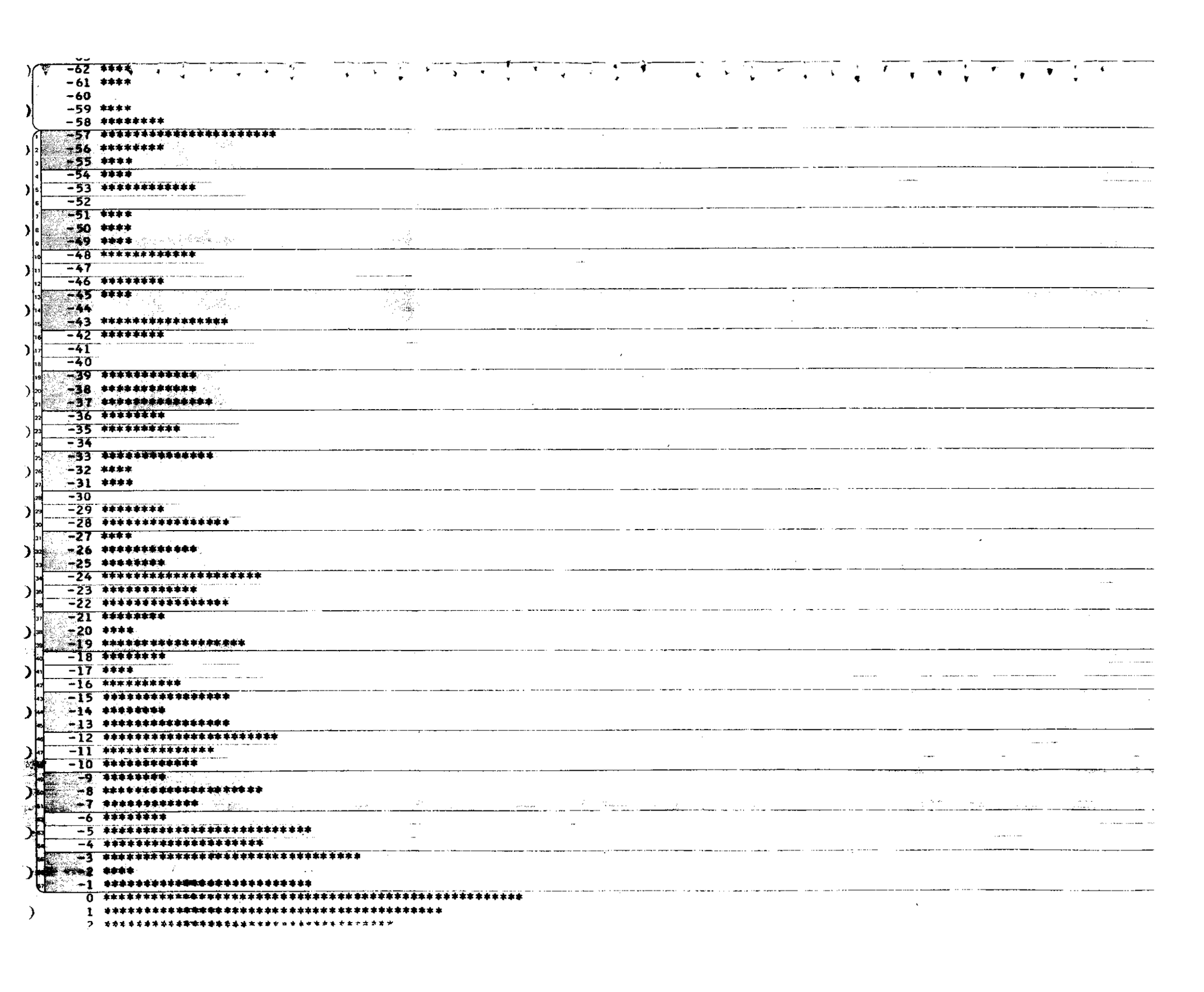
2	5	11	-2	-3	93
-24	52	52	-80	-26	-37
-71	-31	48	0	-167	-166
-35	-59	-48	0	-78	-45
43	0	-13	-66	18	169
110	3	59	31	-149	-57
66	-57	-141	-90	-39	68
-56	-36	-110	-10	168	0
-7	-13	-9	-32	-191	-53
5	-109	17	-16	-88	1
91	-82	-19	-24	37	-28
-1	21	-146	12	-137	46
31	22	-7	-244	92	125
-27	44	-71	-148	26	65
30	195	0	32	-3	-62
-56	-50	-4	21	13	-3
56	94	48	5	-53	16
-5	162	-5	-140	-102	6
-71	170	-20	8	-130	194
-4	-216	80	69	-29	-46
103	126	-87	7	10	-29
-206	-15	1	-43	122	24
145	-23	1	-3	-8	209
-8	-43	41	-13	95	178
8	-39	-49	-179	236	-1
151	-22	1	107	18	52
8	-25	-25	-23	-4	5
17	5	14	-110	58	-64
77	4	15	76	-24	-108
25	-187	59	-14	12	51
-57	174	-155	124	-213	-210
7	64	22	27	-48	-22
-78	2	-6	-18	-104	-36
-19	-22	-1	-58	-110	-12
7	29	88	-189	-3	45
51	20	24	1	246	-61
-3	18	0	-21	-12	2
0	-1	29	-211	12	112
95	-39	-85	-196	88	11
-14	95	-99	-15	-211	-24
6	-53	94	-81	-22	-48
15	36	-6	116	-5	-11
4	5	184	144	-215	1
34	36	2	10	21	2
-5	1	82	-113	-35	-11
-78	-150	31	-1	26	2
177	249	162	25	52	-37
-80	41	2	83	40	163
-75	41	-96	68	12	-77
1	16	-33	-16	18	112
128	-165	-67	136	9	-107
30	-57	30	-207	-139	-134
37	50	-12	-19		

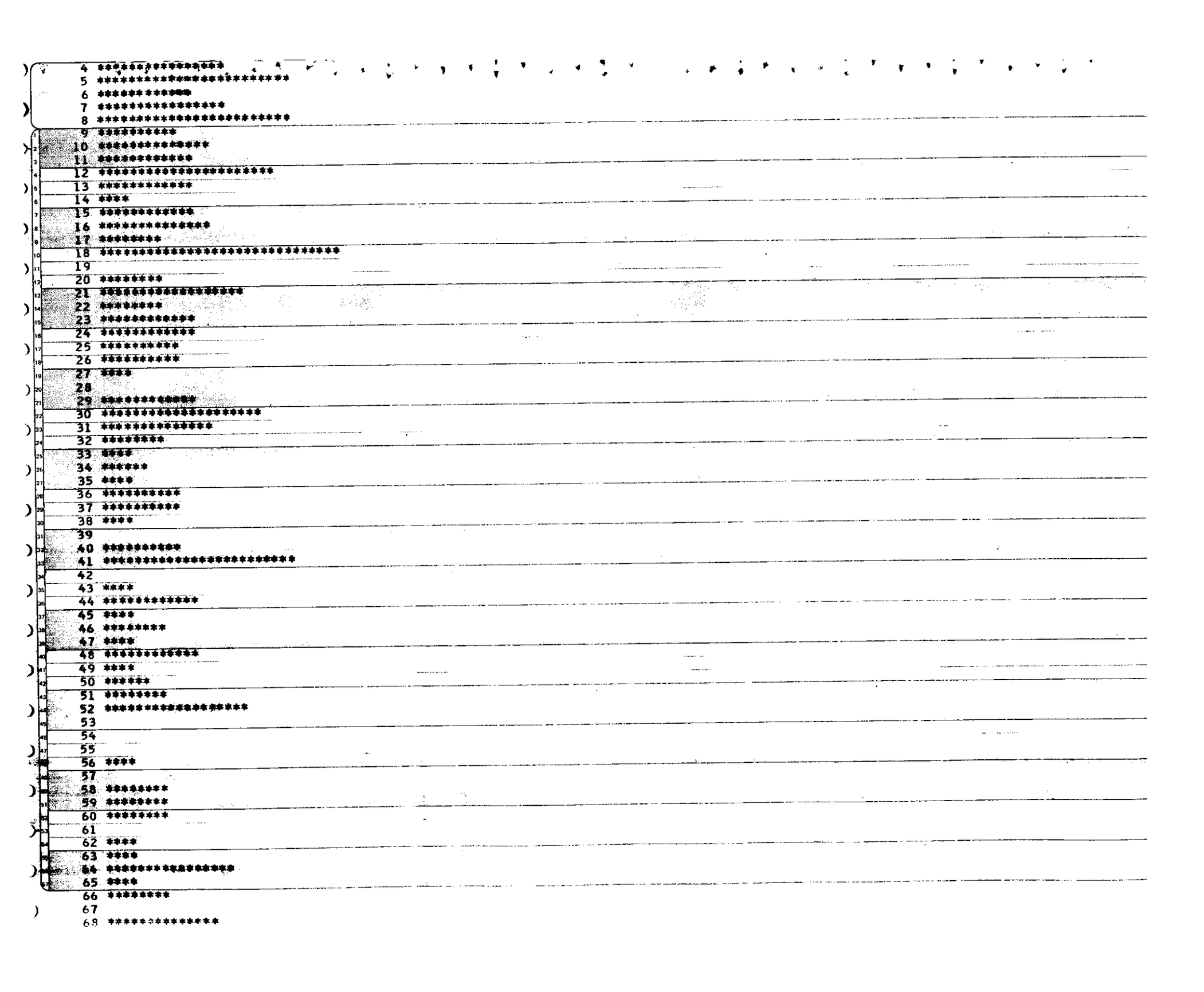
DISTRIBUTION HISTOGRAPH

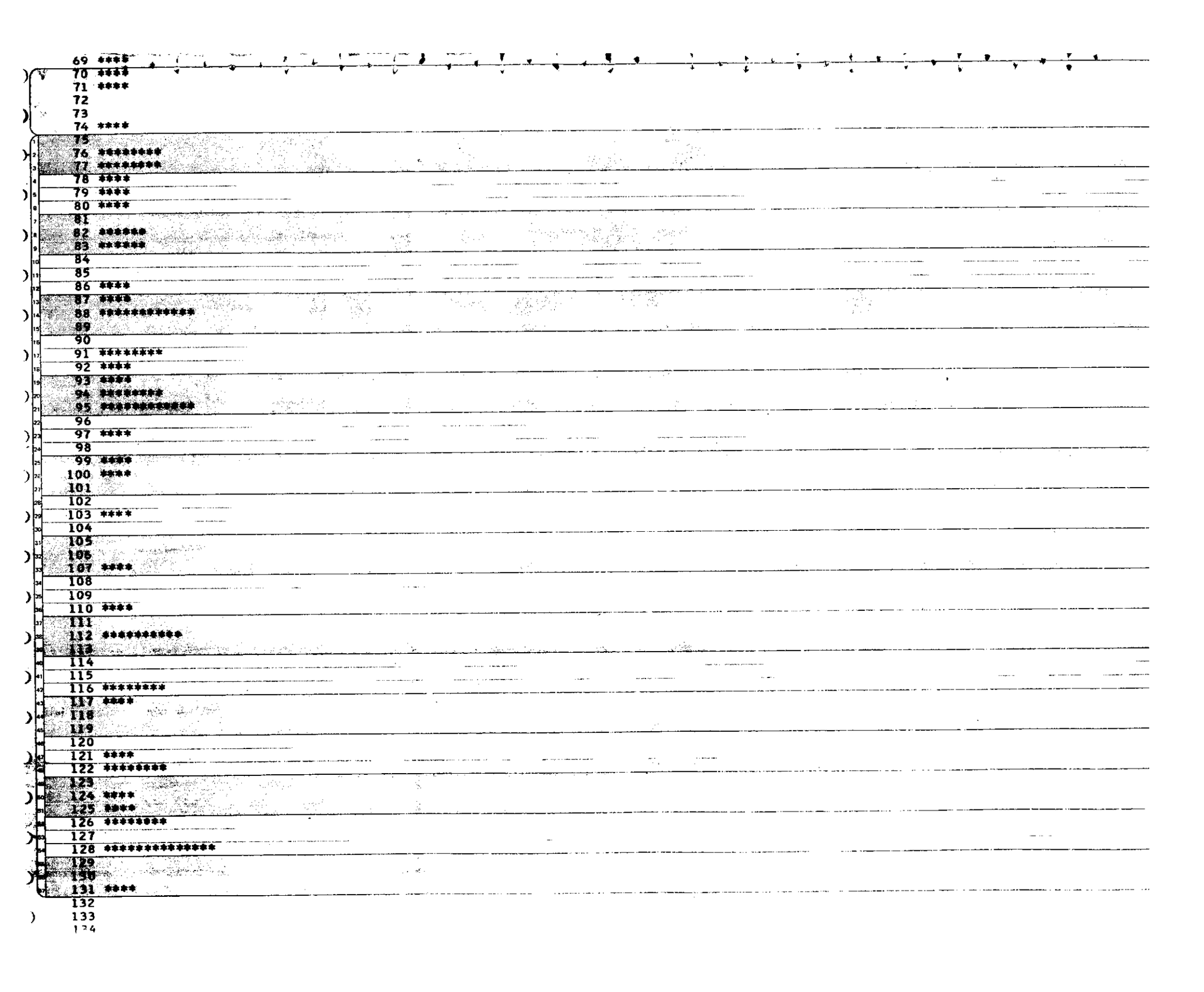
```
-255
-254
-253
-252
-251
-250
-249 ****
-248
-247
-246
-245
-244 ****
-243
-242
-241
-240
-239
-238
-237
-236
-235
-234
-233
-232
-231
-230
-229
-228 ********
-227
-226
-225
-224
-223
-222
-221
-220
-219
-218
-217
-216 ********
-215 ******
-214
-213 ****
-212
-211 ********
-210 ****
-209
-208
-207 ******
-206 ****
-205
-204
-203
-202
-201
-200
-199
-198 ****
```

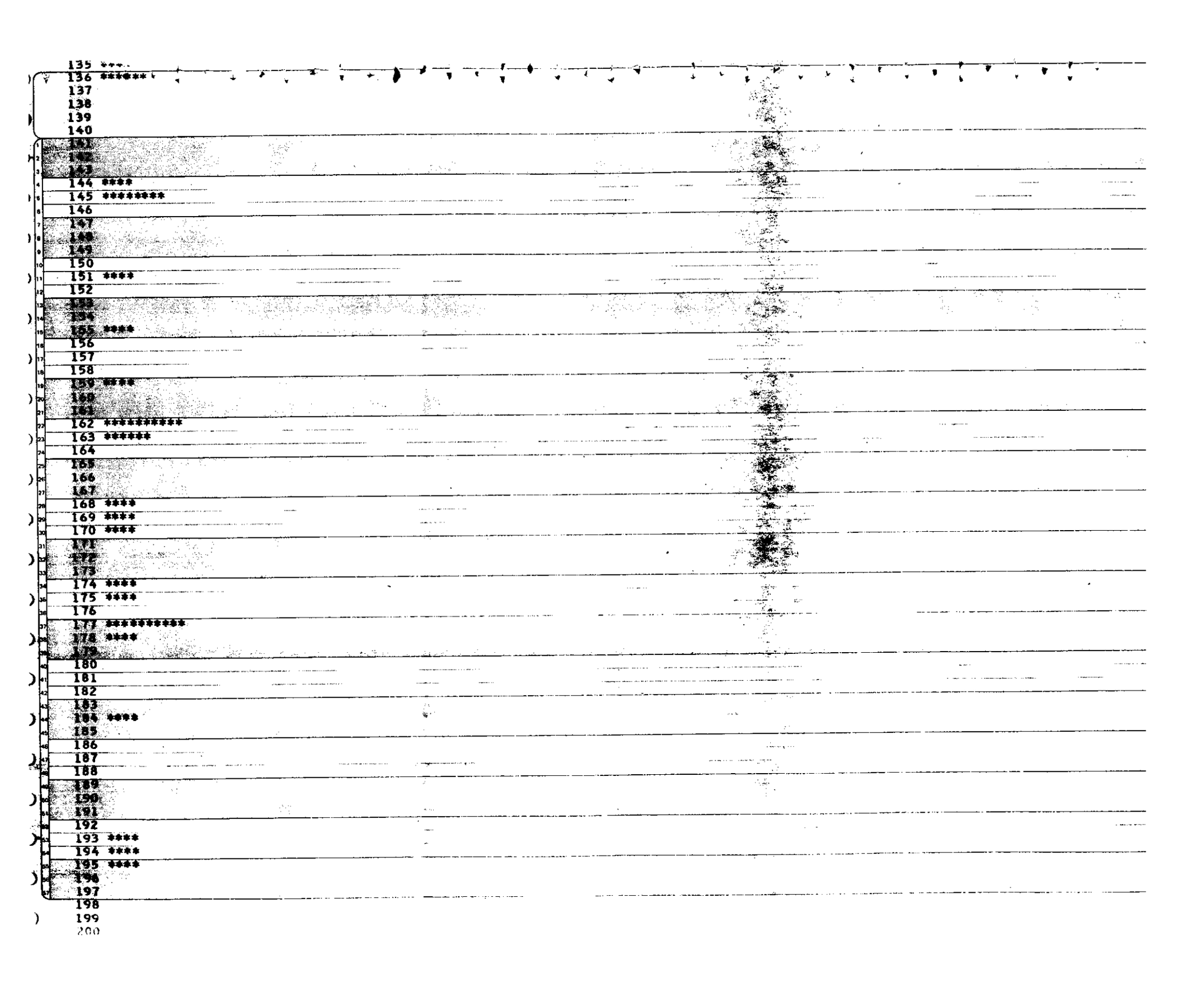

-194
-193
-192
-191 ****
-190
-189 ****
-188
-187 ********
-186
-185
-184
-183
-182
-181
-180 ****
-179 ****
-178
-177
-176
-175
-174
-173
-172
-171
-170
-169
-168
-167 ****
-166 ****
-165 ******
-164
-163
-162
-161
-160
-159
-158
-157
-156
-155 ****
-154
-153
-152
-151
-150 ******
-149 ****
-148 ********
-147
-146 ****
-145 ********
-144
-143
-142
-141 ****
-140 ****
-139 ******
-138
-137 ****
-136
-135
-134 ******
-133
-132
-131
-130 ****


-128 ****
-127 ****
-126 ****
-125
-124
-123
-122
-121
-120
-119
-118 ********
-117
-116
-115
-114
-113 ******
-112
-111 ****
-110 **********
-109 ****
-108 ********
-107 **********
-106
-105 ****
-104 ****
-103
-102 ****
-101 ****
-100
-99 **********
-98
-97 ****
-96 ******
-95
-94
-93
-92
-91
-90 ****
-89 ****
-88 ********
-87 ****
-86
-85 ****
-84
-83 ****
-82 ****
-81 ****
-80 **********
-79
-78 ##################
-77 ******
-76
-75 ##############
-74
-73
-72
-71 ##########
-70
-69 ****
-68
-67 **********
-66 ****
-65
-64 ********


-62 ****
-61 ****
-60
-59 ****
-58 ********
-57 **********
-56 ********
-55 ****
-54 ****
-53 **********
-52
-51 ****
-50 ****
-49 ****
-48 **********
-47
-46 ********
-45 ****
-44
-43 **********
-42 ********
-41
-40
-39 **********
-38 **********
-37 **********
-36 ********
-35 **********
-34
-33 **********
-32 ****
-31 ****
-30
-29 ********
-28 **********
-27 ****
-26 **********
-25 ********
-24 **********
-23 **********
-22 **********
-21 ********
-20 ****
-19 **********
-18 ********
-17 ****
-16 **********
-15 **********
-14 ********
-13 **********
-12 **********
-11 **********
-10 **********
-9 ********
-8 **********
-7 **********
-6 ********
-5 **********
-4 **********
-3 **********
-2 ****
-1 **********
0 **********
1 **********
2 **********







```
201
202
203
204
205
206
207
208
209 ****
210
211
212
213
214
215
216
217
218
219 ****
220
221
222
223 ****
224
225
226
227
228
229
230
231
232
233
234
235
236 ****
237 ****
238
239
240
241
242
243
244
245
246 ****
247
248
249 ******
250
251
252
253
254
255
```

LINES(0)= 2.600000E+01;

CONVERSION TO HEXADECIMAL

D7	01	22	24	02	0A
15	02	05	01	52	71
23	0B	4E	96	1F	01
1A	02	81	F9	A2	19
34	25	50	29	02	53
28	A3	4B	29	60	44
0C	4D	01	10	21	10
12	70	80	A5	43	88
09	6B	1E	39	1E	CF
88	86	25	32	0C	13
83	C6	0C	20	76	B4
28	0B	91	5B	08	40
DF	21	03	40	0D	F9
08	D8	1C	ED	4E	17
21	31	18	17	58	07
26	63	45	1D	2A	15
2F	12	91	08	3A	1C
12	65	69	9B	2E	46
43	C1	11	6F	1A	80
07	3F	40	2E	3C	01
8B	4A	05	26	2B	B1
0B	26	33	08	21	00
05	17	61	29	01	36
79	7A	74	14	3A	59
26	63	0F	4B	04	0D
42	00	76	0C	04	12
7E	00	08	3E	6C	E4
25	1E	13	2C	28	48
0C	80	3C	91	00	87
06	1A	4C	04	53	12
9F	58	80	17	10	64
2A	94	0D	29	40	09
E4	6B	03	0A	37	56
7F	04	61	4F	47	18
4D	57	00	30	39	0F
15	AF	0A	44	2C	08
00	75	0F	09	4E	23
7E	1C	08	0A	02	05
0B	02	03	5D	18	34
34	50	1A	25	47	1F
30	00	A7	A6	23	3B
30	00	4E	2D	2B	00
0D	42	12	A9	6E	03
3B	1F	95	39	42	39
8D	5A	27	44	38	24
6E	0A	A8	00	07	0D
09	20	BF	35	05	6D
11	10	58	01	5B	52
13	18	25	1C	01	15
92	0C	89	2E	1F	16
07	F4	5C	7D	1B	2C
47	94	1A	41	1E	C3
00	20	03	3E	38	32
04	15	0D	03	38	5E
30	05	35	10	05	A2
05	8C	66	06	47	AA
14	08	82	C2	04	D8
50	45	1D	2E	67	7E

02	05	08	02	03	5D
18	34	34	50	1A	25
47	1F	30	00	A7	A6
23	3B	30	00	4E	2D
28	00	0D	42	12	A9
0E	03	38	1F	95	39
42	39	8D	5A	27	44
38	24	6E	0A	A8	00
07	0D	09	20	BF	35
05	6D	11	10	58	01
58	52	13	18	25	1C
01	15	92	0C	89	2E
1F	16	07	F4	5C	7D
18	2C	47	94	1A	41
1E	C3	00	20	03	3E
38	32	04	15	0D	03
38	5E	30	05	35	10
05	A2	05	8C	66	06
47	AA	14	08	82	C2
04	D8	50	45	1D	2E
67	7E	57	07	0A	1D
CE	0F	01	2B	7A	18
91	17	01	03	08	D1
08	2B	29	00	5F	82
08	27	31	B3	EC	01
97	16	01	6B	12	34
08	19	19	17	04	05
11	05	0E	6E	3A	40
4D	04	0F	4C	18	6C
19	B8	3B	0E	0C	33
39	AE	9B	7C	D5	D2
07	40	16	1B	30	16
4E	02	06	12	68	24
13	16	01	3A	6E	0C
07	1D	58	BD	03	2D
33	14	18	01	F6	30
03	12	00	15	0C	02
00	01	1D	D3	0C	70
5F	27	55	C4	58	08
0E	5F	63	0F	D3	18
06	35	5E	51	16	30
0F	24	06	74	05	08
04	05	B8	90	D7	01
22	24	02	0A	15	02
05	01	52	71	23	0B
4E	96	1F	01	1A	02
B1	F9	A2	19	34	25
50	29	02	53	28	A3
48	29	60	44	0C	4D
01	10	21	10	12	70
80	A5	43	88	09	68
1E	39	1E	CF	8B	86
25	32	0C	13		

IN STMT 52 PROGRAM RETURNS FROM MAIN PROCEDURE

```
 ALL ACTIVE BLOCKS AND SCALAR AUTOMATIC VARIABLES
  BLOCK #    1 (MAIN PROCEDURE)

 SUM_OF_THE_SQUARES=    7557763                HEXVALUE='13'          Y= 0.2988          X= 0.8447          J=      3
 I=    1025              STD_DEV=      85.940  MEAN=      -1.432

 LABEL/ENTRY STMT COUNT   LABEL/ENTRY STMT COUNT
 NORMAL      0043 01092   SAMPLES     0001 00001

 *STMT* DYNAMIC FLOW TRACE
 *0052*          0042->0033   0012*(0035->0034)          0034->0036   0013*(0039->0038)          0038->0040          0042->0033
 *0052*  0015*(0035->0034)          0034->0036   0003*(0039->0038)          0038->0040          0042->0033   0014*(0035->0034)
 *0052*          0034->0036   0012*(0039->0038)          0038->0040          0042->0033          0033->0043          0052->0000

  CORE USAGE(BYTES): SYMBOL TABLE 2876, OBJECT CODE 4432, STATIC AND EXTERNAL STORAGE 0000, AUTOMATIC STORAGE     13486, UNUSED
      43168.

 COMPILE TIME   2.96 SECONDS.

 EXECUTION TIME 269.13 SECONDS.
```

57	07	0A	1D	CE	0F
01	28	7A	18	91	17
01	03	08	D1	08	28
29	0D	5F	B2	08	27
91	B3	EC	01	97	16
01	68	12	34	08	19
19	17	04	05	11	05
0E	6E	3A	40	4D	04
0F	4C	18	6C	19	BB
38	0E	0C	33	39	AE
98	7C	D5	D2	07	40
16	18	30	16	4E	02
06	12	68	24	13	16
01	3A	6E	0C	07	1D
58	BD	03	2D	33	14
18	01	F6	3D	03	12
00	15	0C	02	00	01
1D	D3	0C	70	5F	27
55	C4	58	08	0E	5F
63	0F	D3	18	06	35
5E	51	16	30	0F	24
06	74	05	0B	04	05
B8	90	D7	01	22	24
02	0A	15	02	05	01
52	71	23	0B	4E	96
1F	01	1A	02	B1	F9
A2	19	34	25	50	29
02	53	28	A3	4B	29
60	44	0C	4D	01	10
21	10	12	70	80	A5
43	88	09	6B	1E	39
1E	CF	8B	86	25	32
0C	13	83	C6	0C	20
76	B4	2B	0B	91	5B
08	40	DF	21	03	40
0D	F9	08	D8	1C	ED
4E	17	21	31	18	17
58	07	26	63	45	1D
2A	15	2F	12	91	08
3A	1C	12	65	69	9B
2E	46	43	C1	11	6F
1A	80	07	3F	40	2E
3C	01	BB	4A	05	26
28	B1	DB	26	33	08
21	00	05	17	61	29
01	36	79	7A	74	14
3A	59	26	63	0F	4B
04	0D	42	00	76	0C
04	12	7E	00	0B	3E
6C	E4	25	1E	13	2C
28	48	0C	80	3C	91
00	87	06	1A	4C	04
53	12	9F	58	80	17
10	64	2A	94	0D	29
40	09	E4	6B	03	0A
37	56	7F	04	61	4F
47	18	4D	57	00	30
39	0F	15	AF	0A	44
2C	08	00	75	0F	09
4E	23	7E	1C	08	0A

CONVERSION TO HEXADECIMAL

2F	AC	12	8E	11	1A
1B	4A	89	B7	07	12
D4	54	05	77	6F	10
31	28	C0	0B	04	1E
37	C9	1E	61	0B	02
0F	27	18	20	62	00
38	36	0D	44	07	0A
27	3C	7F	76	18	3A
68	D8	AF	65	1C	1B
1C	43	2D	36	1F	8A
1E	00	06	00	52	4C
11	2F	0D	01	83	2D
09	A5	27	1F	09	02
59	07	6C	0D	0C	47
50	0C	26	0A	25	57
19	68	35	5C	14	3D
47	D7	62	47	27	5A
15	24	02	0A	94	19
18	23	1C	08	05	81
88	0A	18	0A	AA	0F
2E	0F	4D	D6	45	33
18	06	80	A5	06	74
1F	44	38	02	23	39
2A	48	9C	43	00	11
8D	CF	36	28	10	EC
38	3E	03	7F	26	0D
1F	50	06	12	0D	06
22	20	08	04	93	15
59	21	58	52	02	05
56	16	11	08	02	92
07	2C	5D	50	21	5C
08	D5	0D	47	81	58
03	14	23	07	57	24
20	0B	53	8E	34	11
1A	2F	23	7E	4E	6C
46	18	51	04	08	67
26	0C	4A	88	C3	87
35	2E	09	D8	00	0A
00	00	08	D6	79	62
18	3F	05	48	27	69
1F	01	7E	95	0F	10
27	36	29	7B	30	BF
3A	0E	0C	33	39	AC
99	7B	03	D0	07	40
16	18	30	16	4E	02
05	12	67	23	13	15
01	39	6D	0C	07	1D
57	BC	03	2D	33	14
18	01	F4	3D	03	12
00	15	0B	02	00	01
1C	D2	08	6F	5E	27
34	C2	57	0B	0E	5F
62	0F	01	18	06	34
5D	50	16	30	0F	23
06	73	05	0B	04	05
87	8E	D5	01	22	23
02	0A	15	02	05	01
51	70	23	0B	40	94

1F	01	1A	02	B0	F7
A1	18	33	25	4F	28
02	52	27	A1	4A	29
5F	44	0C	4D	01	10
21	10	12	6F	7F	A4
43	87	09	68	1E	39
1E	CE	8A	85	25	31
0B	13	82	C5	0C	20
75	B2	28	0B	90	5A
08	3F	0D	21	03	40
0D	F7	08	D7	1C	EB
4D	17	21	31	18	17
57	07	26	62	45	1D
2A	14	2E	12	90	08
3A	1C	12	64	68	99
2E	45	42	BF	11	6E
1A	7F	07	3E	40	2E
3C	01	89	4A	05	26
2B	AF	D9	25	33	08
21	00	05	17	61	28
01	35	78	79	73	13
39	58	26	62	0E	48
04	0D	42	00	75	0B
04	11	7D	00	0A	3D
6B	E2	25	1E	13	2C
28	4A	0C	7E	3C	90
00	86	06	1A	48	04
52	12	9E	57	7E	17
10	63	29	93	0C	29
3F	09	E2	6A	03	0A
37	56	7E	04	60	4F
47	17	4D	57	00	2F
39	0F	15	AE	0A	44
2C	08	00	74	0F	09
4D	23	7D	1C	08	0A
02	05	0A	02	03	5D
18	33	33	50	19	24
46	1F	30	00	A6	A5
23	3B	2F	00	4D	2D
28	00	0D	42	12	A8
6D	03	3A	1E	94	39
42	38	8C	59	27	44
37	24	6D	0A	A6	00
07	0D	09	20	BE	34
05	6C	11	10	58	01
5A	52	13	18	25	1C
01	14	91	0C	88	2E
1F	15	07	F2	5B	7C
1B	2B	46	92	1A	40
1E	C2	00	1F	03	3D
37	32	04	15	0D	02
38	5E	30	05	35	10
05	A1	05	8B	65	06
47	A9	14	08	81	C0
04	D6	50	44	1C	2E
66	7D	56	07	0A	1D
CC	0F	01	2B	79	18
90	17	01	03	08	CF
08	2A	29	0D	5E	B1
08	26	31	B1	EA	01

96	16	01	6A	12	34
08	19	18	17	04	05
11	05	0E	6D	3A	3F
4D	04	0F	4C	18	6B
19	B9	3A	0B	0C	33
39	AC	99	7B	D3	D0
07	40	16	1B	30	16
4E	02	05	12	67	23
13	15	01	39	6D	0C
07	1D	97	8C	03	2D
33	14	18	01	F4	3D
03	12	00	15	08	02
00	01	1C	D2	0B	6F
5E	27	54	C2	57	0B
0E	5F	62	0F	D1	18
06	34	5D	90	16	30
0F	23	06	73	05	0B
04	05	B7	8E	D5	01
22	23	02	0A	15	02
05	01	51	70	23	0B
4D	94	1F	01	1A	02
B0	F7	A1	18	33	25
4F	28	02	52	27	A1
4A	29	5F	44	0C	4D
01	10	21	10	12	6F
7F	A4	43	87	09	6B
1E	39	1E	CE	8A	85
25	31	0B	13	82	C5
0C	20	75	B2	28	0B
90	5A	08	3F	DD	21
03	40	0D	F7	08	D7
1C	EB	4D	17	21	31
18	17	57	07	26	62
45	1D	2A	14	2E	12
90	08	3A	1C	12	64
68	99	2E	45	42	8F
11	6E	1A	7F	07	3E
40	2E	3C	01	B9	4A
05	26	28	AF	D9	25
33	08	21	00	05	17
61	28	01	35	78	79
73	13	39	58	26	62
0E	48	04	0D	42	00
75	0B	04	11	7D	00
0A	30	6B	E2	25	1E
13	2C	28	4A	0C	7E
3C	90	00	86	06	1A
48	04	52	12	9E	57
7E	17	10	63	29	93
0C	29	3F	09	E2	6A
03	0A	37	56	7E	04
60	4F	47	17	4D	57
00	2F	39	0F		

IN STMT 52 PROGRAM RETURNS FROM MAIN PROCEDURE

```
   DO I=1 TO UNITS;
     GET LIST(VI(I),SD(I));
     END;

 /* READ IN THE TOTAL NUMBER OF ITERATIONS (NUMBER OF SAMPLES)                */
   GET LIST(ITER);


 /* THIS IS THE MAIN LOOP--THE LOOP THAT CONSUMES MOST OF THE CPU TIME.       */
   DO J=1 TO ITER;

 /* I CONTAINS THE UNIT (PARAMETER PAIR) THAT THIS ITERATION WILL GENERATE    */
 /* A VALUE FOR . I IS RANDOM VARIABLE FROM 1 TO THE VALUE OF UNITS.          */
     I = 1 + UNITS * RAND(X);
     X = RAND(X);

     T = VALUES(POINTR) * 3 * SD(I) / 256;

 /* MAINTAIN THE POINTR OFFSET IN THE SAMPLE DATA BASE. */
     IF POINTR = 0 THEN POINTR = HBOUND(VALUES,1);
                   ELSE POINTR = POINTR - 1;

     T = VI(I) + T * RNDSGN(B);
     IF T > 255 THEN T = 255;
     IF T < 0 THEN T = 0;

 /* "ROLL" THE SIMULATED BITS OF RNDSGN TO THE LEFT A BIT AT A TIME.          */
     IF B = 32 THEN B = 1;
               ELSE B = B + 1;

 /* RECORD STATISTICAL DATA FOR THIS ITERATION.                               */
     TIMES(I) = TIMES(I) + 1;
     SUM(I) = SUM(I) + T;
     SUMSQ(I) = SUMSQ(I) + T ** 2;

     END;


 /* "ON CONDITION" BLOCK--PRINTS HEADING AT TOP OF PAGE.                      */
   ON ENDPAGE(SYSPRINT) BEGIN;
     IF FIRST = 0 THEN PUT PAGE; FIRST = 0;
     PUT EDIT('GENERATED', 'GIVEN', 'CALC', 'GIVEN', 'CALC')
       (COL(14),A,COL(38),A,COL(51),A,COL(85),A,COL(98),A);
     PUT SKIP EDIT('CASE', 'SAMPLES', 'VI', 'MEAN', '% ERROR', '1SD','1SD',
       '% ERROR')
       (COL(4),A,COL(15),A,COL(40),A,COL(51),A,COL(64),A,COL(86),A,COL(99),A,
       COL(110),A); PUT SKIP(2);
     END;


   PUT PAGE EDIT('*** M6800 ALC DEVIAT SR SIMULATION ***')(COL(47),A);
   PUT SKIP(3) EDIT('TOTAL ITERATIONS:', ITER)(A,F(7));
   PUT SKIP EDIT('SHARING CASES:',UNITS)(A,F(7)); PUT SKIP;
   SIGNAL ENDPAGE(SYSPRINT);

 /* CALCULATE RESULTS OF EACH ITERATION AND PRINT RESULTS.                    */
   DO I=1 TO UNITS;

     MEAN = FLOAT(SUM(I)) / FLOAT(TIMES(I));
     DEV = SQRT((SUMSQ(I) - TIMES(I) * MEAN ** 2) / (TIMES(I) - 1));

     PUT SKIP EDIT(I,TIMES(I),VI(I),MEAN)(F(7),COL(15),F(7),COL(39),F(3),
       COL(48),F(10,6));

     IF VI(I) = THEN ERROR = 0;
```

Testing of

The VI Schedule of Reinforcement Program

prepared for

Dr. Daniel W. Scott III

CSCI 488

by

Daniel Paul Long

Sept. 4, 1979

Testing of
The VI Schedule of Reinforcement Program

This paper discusses the test rationales and results associated with testing the VI Schedule of Reinforcement Program (VISORP). See the paper: Functional Description of the VI Schedule of Reinforcement Program, for information about the program itself. The following is organized into three parts: pre-session, post-session, and session. The discussion of pre-session concerns the testing of the parts of the program performed before the VI schedule is executed (ie. session set up, eg. initialization), the post-session concerns the testing of the parts of the program performed after the VI scedule is executed (eg. output results), and the session deals with the testing of the parts of the program that perform the execution of the VI schedule.

PRE-SESSION

To test the initialization operation performed before each session, I executed the subroutines that do the initialization and then observed the results by using the monitor's (SMARTBUG) memory examine function. I observed that the initializations--zeroing a section of lower memory for variable work space, assigning default parameters (eg. length of session), initializing each subject's timer (clock), and configuring the interfaces--were performed as expected. (The timer initialization is done after the

user has defined the time interval characteristics from the terminal; the other initializations are done immediately upon execution of VISORP.)

The parameter definition operation was tested in a similar manner. I executed the main subroutine responsible for accepting data from the terminal (this subroutine is called INPUT), supplying a variety of input values and combinations of values, and then observing the contents of memory to see if they agreed with what I had entered. INPUT detected and indicated out-of-range values and invalid characters, performed the required base conversion (base 10 to base 2) on numeric data, and stored all data in the appropriate locations in memory. I was also able to direct the flow of control from INPUT (restart INPUT, end VISORP, or execute a session) with no trouble.

POST-SESSION

The post-session operation--mainly the displaying of session results--was tested initially by setting the contents of memory where the results would be accumulated to various values using the monitor's memory change function, executing the principle subroutine responsible for output (called OUTPUT), and observing the displayed values on the CRT to compare them with the values I entered. OUTPUT retrieved all data (accumulated responses and reinforcements for each subject) from the appropriate locations in memory,

performed the required base conversion (base 2 to base 10), and displayed the results in the designed tabular form. The displayed values agreed with the corresponding contents of memory. I was later able to use data generated by a manually simulated session instead of directly setting the contents of memory using the monitor. "Real" data, of course, did not result in modified (or improved) performance from the OUTPUT subroutine--I just felt increased satisfaction with the new evidence that the entire system worked together.

SESSION

The part of VISORP that performs the session schedule was tested by connecting the feeder mechanism's coil and the metal bar's switch of a rat chamber to the PIO board (see the paper: The Parallel Input/Output Board). During execution of a session, I would manipulate the metal bar in the chamber and listen for the coil to fire, indicating a reinforcement had been given.

If I had entered 0 for the unit's VI (mean time interval) and any value for the dispersion, the coil would fire immediately after each press on the bar, with no apparent time interval between the two actions (continuous reinforcement or CRF). When I would enter a non-zero VI (eg. 10) and a dispersion (SD) of zero, with a rate of response (pressing bar) of about 3 Hz, the coil would fire at regular intervals

according to the previously entered VI--ten seconds between firings for a VI of 10. This is called a fixed interval schedule of reinforcement. I ran several sessions with non-zero VI's and dispersions. For instance, with a VI of 30 seconds and one standard deviation from the mean (SD) of 15 seconds, while maintaining the same 3 Hz rate of response manually, the coil would fire sometimes before and sometimes after 30 seconds had elapsed between firings. As I decreased the SD value in subsequent sessions, the firings tended to occur closer to the VI (mean time interval), while increasing the SD value resulted in the coil tending to fire with greater deviation from the VI.

All of the above was done with one rat chamber interfaced to one of the eight I/O channels. I performed trials using each of the eight channels separately with consistently similar results. Testing more than one chamber using VISORP at one time was immpossible since I only had one rat chamber available. To simulate more than one chamber, I set up VISORP to monitor, for instance, four chambers, and by moving the I/O wires, which are between the PIO board and chamber, first to one channel and then to another on the PIO board, I "faked out" the program into believing there were four chambers using VISORP. I performed this simulation for several combinations of parameters for each chamber as I had for just the one chamber described above. As far as I could tell, each chamber

was serviced separately by VISORP--the responses and reinforcements occuring in one chamber did not effect another chamber's service.

When the coil fired up to the maximum number of reinforcements allowed, specified by the user before the session, the proper LED on the PIFG/AVI board (see the paper: The Presettable Interrupt Frequency Generator/ Audio-Visual Indicator Board) would light up, indicating which unit, or chamber, had reached its limit, and the piezo-electric transducer would beep.

The heart of VISORP, and what made it so much harder to write than a simpler schedule of reinforcement (eg. fixed interval), is the subroutine called DEVIAT. This subroutine returns random, normally distributed values according to a given mean and dispersion passed into it. Up till now, I have described how I tested VISORP without attempting to analyze the time intervals produced in VISORP via DEVIAT. I merely confirmed by observation, that, basically, the program was working with no blatant flaws in its time interval generation and overall logic. To analyze DEVIAT more closely, I simulated its use in VISORP with two PL/I programs running on an IBM 360/50.

The first program, called NORMTEST, simulates a total of 936 different combinations of VIs and SDs, ranging from 0 to 255 for VI and 85 for SD, both in increments of 5. Each case is a sample run with unique parameters. NORMTEST may be used to observe trends in error, such as

the increase in error of VI as VI decreases. To find the error produced by a particular pair of parameters, say VI of 30 and SD of 15, look for the VI in the column labeled GIVEN VI. Next, look over to the left under the column labeled GIVEN 1SD for the SD. This line will have the % error for the VI and SD. In the above case, the VI and SD error is shown to be .1284% and 7.0886%, respectively. Note that these errors are not exactly what you should expect when using the DEVIAT subroutine--they are examples, or ball park figures of what you may experience. NORMTEST is designed to give you an idea of the order of magnitude of error for particular VI's and SD's produced by DEVIAT.

The second program, called DISTRIB, produces a histograph of the occurrence of time intervals for a given VI and SD. I have included a few printouts to graphically show the distibution. The "piling up" near 0 and 255 is more abrupt than I had anticipated, though overall, the distributions are what I expected.

NORMTEST

program listing and output

prepared for

Dr. Daniel W. Scott III
CSCI 488

by

Daniel Paul Long

```
/*     NORMTEST     */

/*     NORMTEST SIMULATES THE USE OF THE SUBROUTINE DEVIAT IN THE PROGRAM      */
                                                                                 */
/*  VISORP.                                                                     */
/*     THIS PROGRAM SIMULATES A TOTAL OF 936 DIFFERENT COMBINATIONS OF VI'S     */
/*  AND SD'S, RANGING FROM 0 TO 255 FOR VI AND 0 TO 85 FOR SD, BOTH IN          */
/*  INCREMENTS OF 5. EACH CASE IS A SAMPLE RUN WITH UNIQUE PARAMEMTERS.         */


TEST:PROC OPTIONS(MAIN);

  DCL X FIXED(9,9),(MEAN,DEV) FIXED(10,6),BYTE CHAR(2),(VALUES(0:1023),
    N INIT(16),RNDSGN(32), ITER, UNITS, I, J, POINTR INIT(1023), T, B
    INIT(1)) BIN FIXED(31), HEX(0:15) CHAR(1) INIT('0','1','2','3','4','5',
    '6','7','8','9','A','B','C','D','E','F'), ERROR FIXED(10,4), FIRST INIT(1);

/* USE THE TIME STRING AS RANDOM SEED X */
  GET STRING(TIME) EDIT(X)(F(9,9));

/* RNDSGN IS AN ARRAY USED TO SIMULATE A FOUR BYTE DATA BASE CONTAINING      */
/* RANDOM 1'S AND 0'S.                                                       */
  GET LIST(RNDSGN);

  PUT LIST('STATIC "VALUES" TABLE WITH 1024 ELEMENTS');
  PUT SKIP LIST('ROW MAJOR ORDER');
  PUT SKIP LIST('FORMAT:   DCM=HX'); PUT SKIP;

  ON ENDPAGE;

/* READ IN THE SAMPLE DATA BASE ONE VALUE AT A TIME AND CONVERT TO BASE 10   */
  DO I=LBOUND(VALUES,1) TO HBOUND(VALUES,1);

    GET LIST(BYTE);

/* FIRST HEX DIGIT */
    DO J=0 TO 15 WHILE (SUBSTR(BYTE,1,1) ¬= HEX(J)); END;
    VALUES(I) = J * 16;

/* SECOND HEX DIGIT */
    DO J=0 TO 15 WHILE (SUBSTR(BYTE,2) ¬= HEX(J)); END;
    VALUES(I) = VALUES(I) + J;

    N = N + 1;
    IF N > 16 THEN DO; N = 1; PUT SKIP; END;
    PUT EDIT(VALUES(I),'=',BYTE)(COL(8 * N - 7),F(3),A(1),A(2));

    END;


  ON ENDPAGE SYSTEM;

  PUT SKIP(3) LIST('SIMULATED BITS IN RNDSGN');
  PUT SKIP(2) EDIT(0.5 - FLOAT(RNDSGN) / 2) (32 F(1));

  GET LIST(UNITS);

/* THE VALUE OF UNITS DETERMINES THE LENGTH OF SEVERAL ARRAYS                */
  DYNAM: BEGIN; DCL (VI(UNITS), SD(UNITS), (TIMES(UNITS), SUM(UNITS),
    SUMSQ(UNITS)) INIT((UNITS)0)) BIN FIXED(31);

/* THE PARAMETERS PAIRS */
```

```
            ELSE ERROR = (1 - MEAN / V$(I)) * 100;
        PUT EDIT(ERROR, SD(I), DEV)(COL(61),F(10,4),
          COL(87),F(2),COL(95),F(10,6));

        IF SD(I) = 0 THEN ERROR = 0;
                     ELSE ERROR = (1 - DEV / SD(I)) * 100;
        PUT EDIT(ERROR)(COL(107),F(10,4));

        END;

      PUT PAGE;

      END DYNAM;

    END TEST;
```

STATIC *VALUES* TABLE WITH 1024 ELEMENTS
ROW MAJOR ORDER
FORMAT: DCM=HX

47=2F	27=1B	212=D4	49=31	55=37	15=0F	56=38	39=27	104=68	28=1C	30=1E	17=11	9=09	89=59	80=50	25=19
71=47	21=15	24=18	139=8B	46=2E	24=18	31=1F	42=2A	141=8D	56=38	31=1F	34=22	89=59	86=56	7=07	8=08
3=03	32=20	26=1A	70=46	38=26	53=35	13=0D	27=1B	31=1F	39=27	58=3A	153=99	22=16	5=05	1=01	87=57
24=18	0=00	28=1C	84=54	98=62	93=5D	6=06	183=B7	2=02	81=51	172=AC	74=4A	84=54	40=28	201=C9	39=27
54=36	60=3C	216=D8	67=43	0=00	47=2F	165=A5	7=07	12=0C	107=6B	215=D7	36=24	35=23	10=0A	15=0F	6=06
68=44	72=48	207=CF	62=3E	80=50	32=20	33=21	22=16	44=2C	213=D5	20=14	11=0B	47=2F	24=18	12=0C	46=2E
0=00	63=3F	1=01	54=36	14=0E	123=7B	27=1B	18=12	87=39	188=BC	1=01	21=15	210=D2	194=C2	15=0F	80=50
115=73	142=8E	10=0A	112=70	18=12	137=89	5=05	192=C0	30=1E	24=18	13=0D	127=7F	175=AF	45=2D	6=06	13=0D
39=27	108=6C	38=26	53=35	98=62	2=02	28=1C	27=1B	77=4D	80=50	56=38	156=9C	54=36	3=03	6=06	8=08
88=58	17=11	93=5D	13=0D	35=23	83=53	35=23	81=51	74=4A	9=09	8=08	5=05	126=7E	41=29	12=0C	211=D3
48=30	103=67	109=6D	3=03	244=F4	11=0B	11=0B	87=57	209=D1	22=16	5=05	213=D5	21=15	35=23	142=8E	183=B7
119=77	11=0B	97=61	32=20	68=44	118=76	101=65	54=36	0=00	1=01	31=1F	13=0D	10=0A	92=5C	71=47	10=0A
11=0B	10=0A	214=D6	165=A5	2=02	67=43	40=28	127=7F	18=12	4=04	82=52	8=08	93=5D	71=47	7=07	142=8E
126=7E	4=04	136=88	216=D8	214=D6	72=48	149=95	123=7B	51=33	208=D0	22=16	35=23	12=0C	45=2D	61=3D	2=02
111=6F	11=0B	24=18	48=30	11=0B	1=01	2=02	11=0B	17=11	7=07	111=6F	4=04	11=0B	98=62	7=07	24=18
28=1C	31=1F	82=52	179=B3	9=09	12=0C	37=25	20=14	39=27	148=94	5=05	170=AA	69=45	6=06	35=23	0=00
29=1D	38=26	13=0D	147=93	2=02	2=02	33=21	33=21	87=57	52=34	78=4E	8=08	195=C3	0=00	121=79	39=27
15=0F	48=30	57=39	7=07	78=4E	19=13	7=07	51=33	3=03	0=00	94=5E	14=0E	6=06	15=0F	4=04	34=22
5=05	77=4D	26=1A	18=12	16=10	30=1E	2=02	0=00	10=0A	58=3A	27=1B	138=8A	76=4C	45=2D	2=02	71=47
87=57	61=3D	90=5A	25=19	129=81	15=0F	51=33	116=74	57=39	17=11	236=EC	13=0D	6=06	21=15	5=05	146=92
92=5C	91=5B	36=24	17=11	108=6C	103=67	135=87	10=0A	98=62	105=69	16=10	191=BF	172=AC	64=40	2=02	21=15
29=1D	20=14	18=12	1=01	39=27	95=5F	52=34	35=23	5=05	35=23	1=01	148=94	31=1F	161=A1	2=02	95=5F
33=21	67=43	30=1E	11=0B	117=75	8=08	13=0D	77=4D	87=57	42=2A	58=3A	46=2E	26=1A	60=3C	43=2B	33=21
1=01	57=39	4=04	4=04	107=6B	40=28	0=00	82=52	16=10	63=3F	55=37	71=47	57=39	44=2C	77=4D	2=02
24=18	70=46	35=23	43=2B	109=6D	66=42	55=37	7=07	5=05	90=5A	1=01	31=1F	27=1B	30=1E	55=37	56=38
5=05	71=47	4=04	102=66	204=CC	144=90	8=08	8=08	1=01	24=18	82=52	68=44	16=10	135=87	206=CE	19=13
178=B2	63=3F	247=F7	23=17	7=07	20=14	28=1C	69=45	127=7F	1=01	175=AF	0=00	53=35	88=58	13=0D	17=11
226=E2	74=4A	134=86	18=12	99=63	9=09	86=56	23=17	15=0F	8=08	35=23	5=05	51=33	31=1F	59=3B	0=00
3=03	56=38	36=24	13=0D	108=6C	82=52	20=14	21=15	43=2B	194=C2	50=32	94=5E	161=A1	169=A9	214=D6	125=7D
15=0F	23=17	42=2A	38=26	26=1A	51=33	39=27	12=0C	18=12	9=09	138=8A	130=82	43=2B	221=DD	8=08	33=21
38=26	46=2E	18=12	66=42	7=07	185=B9	217=D9	5=05	120=78	38=26	66=42	125=7D	37=25	12=0C	6=06	158=9E
41=29	226=E2	126=7E	77=4D	21=15	0=00	125=7D	10=0A	51=33	48=30	47=2F	13=0D	58=3A	140=8C	109=6D	9=09
17=11	19=13	145=91	7=07	70=46	0=00	4=04	48=30	5=05	20=14	80=50	86=56	1=01	1=01	41=29	49=31
2=02	37=25	161=A1	77=4D	111=6F	107=6B	133=85	197=C5	11=0B	33=21	215=D7	49=31	98=62	18=12	100=64	191=BF
62=3E	74=4A	37=25	23=17	121=79	98=62	0=00	0=00	30=1E	126=7E	26=1A	87=57	147=93	106=6A	4=04	87=57
174=AE	116=74	28=1C	2=02	80=50	0=00	0=00	66=42	30=1E	89=59	10=0A	32=20	16=10	24=18	12=0C	242=F2
146=92	31=1F	21=15	5=05	139=8B	8=08	68=44	7=07	43=2B	3=03	13=0D	177=B1	176=B0	79=4F	74=4A	1=01
127=7F	30=1E	37=25	12=0C	144=90	3=03	28=1C	24=18	69=45	144=90	104=68	17=11	64=40	5=05	51=33	97=61
115=73	14=0E	117=75	10=0A	19=13	60=3C	75=4B	126=7E	12=0C	3=03	96=60	0=00	10=0A	15=0F	8=08	3=03
25=19	166=A6	77=4D	18=12	148=94	39=27	166=A6	190=BE	88=58	37=25	136=88	91=5B	26=1A	3=03	61=3D	53=35
101=65	125=81	28=1C	10=0A	121=79	8=08	94=5E	234=EA	247=F7	40=28	41=29	16=10	164=A4	57=39	49=31	32=20
90=5A	64=40	235=EB	23=17	29=1D	8=08	153=99	110=6E	46=2E	38=26	8=08	40=28	19=13	75=4B	11=0B	61=3D
44=2C	144=90	4=04	23=17	41=29	10=0A	79=4F	47=2F	68=44	9=09	10=0A	93=5D	36=24	165=A5	45=2D	168=A8
57=39	68=44	0=00	52=34	1=01	28=1C	46=2E	124=7C	64=40	61=3D	2=02	16=10	6=06	192=C0	46=2E	29=1D
24=18	207=CF	177=B1	1=01	150=96	8=08	17=11	77=4D	25=19	57=39	7=07	78=4E	19=13	7=07	51=33	3=03
0=00	94=5E	14=0E	6=06	15=0F	4=04	34=22	5=05	77=4D	176=B0	79=4F	74=4A	1=01	127=7F	30=1E	37=25
12=0C	144=90	3=03	28=1C	24=18	69=45	144=90	104=68	17=11	64=40	5=05	51=33	97=61	115=73	14=0E	117=75
10=0A	19=13	60=3C	75=4B	126=7E	12=0C	3=03	96=60	0=00	22=16	25=19	5=05	4=04	185=B9	172=AC	64=40
2=02	21=15	29=1D	20=14	18=12	1=01	39=27	95=5F	52=34	35=23	5=05	35=23	1=01	148=94	247=F7	40=28
41=29	16=10	164=A4	57=39	49=31	32=20	90=5A	64=40	235=EB	23=17	29=1D	8=08	153=99	110=6E	46=2E	38=26
8=08	40=28	19=13	75=4B	11=0B	61=3D	44=2C	144=90	4=04	23=17	41=29	10=0A	79=4F	47=2F	1=01	24=18
14=0E	15=0F	58=3A	153=99	22=16	5=05	1=01	87=57	24=18	0=00	28=1C	84=54	98=62	93=5D	6=06	183=B7
2=02	81=51	31=1F	161=A1	2=02	95=5F	33=21	67=43	30=1E	11=0B	117=75	8=08	13=0D	77=4D	87=57	42=2A
58=3A	46=2E	26=1A	60=3C	43=2B	33=21	1=01	59=3B	4=04	4=04	107=6B	40=28	0=00	82=52	16=10	63=3F
55=37	71=47	57=39	106=6A	23=17	109=6D	76=4C	14=0E	123=7B	27=1B	18=12	57=39	188=BC	1=01	21=15	210=D2
194=C2	15=0F	80=50	115=73	142=8E	10=0A	112=70	1=01	24=18	82=52	68=44	16=10	135=87	206=CE	19=13	178=B2
63=3F	247=F7	23=17	7=07	20=14	28=1C	69=45	127=7F	1=01	175=AF	0=00	53=35	88=58	13=0D	17=11	226=E2
74=4A	134=86	18=12	99=63	9=09	86=56	23=17	15=0F	18=12	4=04	58=3A	24=18	12=0C	211=D3	48=30	103=67

18=12	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	8=08	33=21	38=26	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
120=78	38=26	66=42	125=7D	37=25	12=0C	6=06	158=9E	41=29	226=E2	126=7E	77=4D	52=34	5=05	63=3F	107=6B
51=33	208=D0	22=16	35=23	12=0C	45=2D	61=3D	2=02	111=6F	11=0B	24=18	48=30	11=0B	1=01	2=02	11=0B
2=02	37=25	161=A1	77=4D	111=6F	107=6B	133=85	197=C5	11=0B	33=21	215=D7	49=31	98=62	18=12	100=64	191=BF
62=3E	74=4A	37=25	23=17	121=79	98=62	0=00	0=00	30=1E	126=7E	26=1A	87=57	147=93	106=6A	4=04	87=57

SIMULATED BITS IN RNDSGN

01010010000011011100101110111001

*** M6800 ALC DEVIAT SR SIMULATION ***

TOTAL ITERATIONS: 936000
SHARING CASES: 936

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
1	1016	0	0.000000	0.0000	0	0.000000	0.0000
2	995	0	1.654271	0.0000	5	3.070117	38.5976
3	1000	0	3.428999	0.0000	10	6.072053	39.2795
4	1005	0	5.449751	0.0000	15	9.062351	39.5843
5	1032	0	6.844961	0.0000	20	11.399678	43.0016
6	971	0	9.377960	0.0000	25	16.172915	35.3083
7	982	0	11.460285	0.0000	30	18.635019	37.8832
8	990	0	13.981818	0.0000	35	22.013776	37.1035
9	995	0	14.966834	0.0000	40	24.021911	39.9453
10	1009	0	15.915758	0.0000	45	27.035651	39.9208
11	987	0	19.098277	0.0000	50	31.829723	36.3406
12	992	0	20.003024	0.0000	55	33.901486	38.3609
13	991	0	19.854692	0.0000	60	34.411430	42.6476
14	1013	0	22.350444	0.0000	65	40.022703	38.4266
15	1024	0	23.468750	0.0000	70	42.025349	39.9638
16	997	0	25.115346	0.0000	75	44.970871	40.0388
17	993	0	28.246727	0.0000	80	48.568630	39.2892
18	987	0	31.292806	0.0000	85	53.466339	37.0985
19	1008	5	5.000000	0.0000	0	0.000000	0.0000
20	950	5	5.367368	-7.3472	5	3.760870	24.7826
21	1028	5	6.998054	-39.9610	10	7.108835	28.9116
22	982	5	8.664969	-73.2993	15	10.635408	29.0973
23	962	5	10.510395	-110.2078	20	13.746871	31.2657
24	1015	5	11.788177	-135.7636	25	16.310760	34.7570
25	1027	5	13.511197	-170.2239	30	18.633184	37.8894
26	1026	5	16.309941	-226.1987	35	23.525294	32.7849
27	976	5	18.088114	-261.7622	40	26.560360	33.5991
28	967	5	18.072388	-261.4477	45	27.421806	39.0626
29	959	5	23.012513	-360.2501	50	33.757421	32.4851
30	1009	5	23.104063	-362.0812	55	36.396734	33.8241
31	959	5	26.461939	-429.2387	60	39.609133	33.9848
32	993	5	26.344410	-426.8881	65	41.839172	35.6321
33	1004	5	26.627490	-432.5497	70	42.306887	39.5614
34	1010	5	31.335643	-526.7127	75	50.480950	32.6921
35	968	5	31.547520	-530.9503	80	52.588664	34.2641
36	1000	5	32.497999	-549.9599	85	53.746672	36.7686
37	1002	10	10.000000	0.0000	0	0.000000	0.0000
38	1048	10	9.987595	0.1240	5	4.467525	10.6495
39	966	10	11.125258	-11.2524	10	8.509541	14.9045
40	980	10	12.147959	-21.4795	15	11.139910	25.7339
41	1013	10	14.023692	-40.2368	20	15.058365	24.7081
42	961	10	16.886576	-68.8656	25	19.352289	22.5909
43	999	10	17.303303	-73.0329	30	20.775571	30.7481
44	1033	10	18.760890	-87.6088	35	24.305587	30.5555
45	1032	10	20.816860	-108.1685	40	27.194567	32.0136
46	992	10	21.842741	-118.4273	45	30.672145	31.8397
47	1000	10	25.252999	-152.5299	50	33.949722	32.1005
48	1029	10	25.994169	-159.9416	55	37.270464	32.2355
49	1020	10	27.002941	-170.0293	60	38.558432	35.7359
50	996	10	30.955823	-209.5582	65	44.136836	32.0972
51	975	10	33.756923	-237.5691	70	48.367533	30.9035

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
52	994	10	32.765593	-227.6558	75	48.170083	35.7732
53	997	10	36.187562	-261.8755	80	54.236485	32.2044
54	1012	10	36.970855	-269.7034	85	57.469892	32.3884
55	1001	15	15.000000	0.0000	0	0.000000	0.0000
56	987	15	15.268490	-1.7898	5	4.576580	8.4684
57	980	15	15.928571	-6.1903	10	8.951407	10.4859
58	1010	15	17.062376	-13.7491	15	12.732401	15.1173
59	1012	15	17.488142	-16.5875	20	15.139733	24.3013
60	1007	15	18.594836	-23.9655	25	18.159682	27.3612
61	982	15	21.715885	-44.7725	30	22.532246	24.8925
62	993	15	23.128902	-54.1926	35	25.619295	26.8020
63	1000	15	24.035999	-60.2399	40	28.699728	28.2506
64	983	15	25.886063	-72.5737	45	30.727652	31.7163
65	1020	15	26.805882	-78.7057	50	34.659051	30.6819
66	1006	15	30.340954	-102.2729	55	38.017907	30.8765
67	1019	15	29.396467	-95.9763	60	40.460267	32.5662
68	950	15	34.147368	-127.6491	65	45.128167	30.5721
69	1036	15	34.199806	-127.9986	70	47.564121	32.0512
70	981	15	37.222222	-148.1480	75	49.972845	33.3695
71	1000	15	39.049999	-160.3332	80	54.641296	31.6984
72	1012	15	40.770750	-171.8049	85	56.842006	33.1271
73	1001	20	20.000000	0.0000	0	0.000000	0.0000
74	992	20	20.141129	-0.7056	5	4.668014	6.6397
75	1008	20	20.542658	-2.7132	10	9.344567	6.5544
76	962	20	20.404365	-2.0217	15	13.557283	9.6181
77	984	20	21.770325	-8.8516	20	15.977352	20.1132
78	1009	20	23.994053	-19.9702	25	19.652486	21.3900
79	972	20	24.803497	-24.0174	30	23.630251	21.2325
80	1005	20	27.062686	-35.3134	35	27.449030	21.5742
81	976	20	27.243852	-36.2192	40	28.872593	27.8185
82	996	20	29.270080	-46.3503	45	32.641933	27.4623
83	997	20	30.948846	-54.7441	50	34.856885	30.2863
84	972	20	33.281893	-66.4093	55	39.201619	28.7243
85	1011	20	35.053412	-75.2670	60	40.608504	32.3191
86	945	20	36.362962	-81.8147	65	44.277899	31.8801
87	980	20	35.966326	-79.8316	70	46.666004	33.3343
88	972	20	41.282921	-106.4146	75	52.356018	30.1919
89	1011	20	41.883283	-109.4163	80	55.352393	30.8095
90	957	20	47.439916	-137.1996	85	59.529236	29.9656
91	994	25	25.000000	0.0000	0	0.000000	0.0000
92	992	25	25.161290	-0.6450	5	4.655672	6.8865
93	1006	25	25.174950	-0.6997	10	9.203137	7.9687
94	1024	25	25.750000	-2.9999	15	14.052965	6.3136
95	993	25	27.112789	-8.4510	20	18.092821	9.5359
96	1000	25	27.566999	-10.2679	25	21.234464	15.0622
97	1041	25	27.663784	-10.6551	30	23.242831	22.5239
98	970	25	29.346391	-17.3854	35	26.231084	25.0540
99	978	25	31.345603	-25.3823	40	30.670988	23.3225
100	1024	25	33.528320	-34.1132	45	33.924290	24.6127
101	991	25	32.866801	-31.4671	50	36.533538	26.9330
102	979	25	38.819203	-55.2767	55	42.045099	23.5544
103	1057	25	39.202459	-56.8098	60	44.160849	26.3986
104	973	25	41.580678	-66.3227	65	47.026160	27.6520
105	998	25	42.344689	-69.3786	70	51.266443	26.7622
106	958	25	42.739039	-70.9561	75	53.334738	28.8870
107	1044	25	45.141762	-80.5669	80	57.566194	28.0423
108	1019	25	45.560353	-82.2413	85	58.812921	30.8083

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
109	1020	30	30.000000	0.0000	0	0.000000	0.0000
110	1005	30	29.910447	0.2985	5	4.805788	3.8843
111	984	30	30.462398	-1.5412	10	9.639621	3.6038
112	986	30	30.038539	-0.1284	15	13.936715	7.0886
113	1001	30	30.982017	-3.2732	20	18.604938	6.9753
114	984	30	32.285569	-7.6185	25	21.807508	12.7699
115	952	30	32.759453	-9.1980	30	24.237693	19.2077
116	1037	30	35.343297	-17.8109	35	29.615226	15.3851
117	1001	30	36.313686	-21.0455	40	31.585052	21.0373
118	1023	30	36.045943	-20.1531	45	34.671690	22.9518
119	1021	30	39.608227	-32.0273	50	38.773263	22.4534
120	1006	30	39.716699	-32.3889	55	41.014186	25.4287
121	951	30	41.302839	-37.6760	60	43.494328	27.5094
122	1020	30	44.431372	-48.1045	65	47.694545	26.6238
123	989	30	43.219413	-44.0646	70	49.363965	29.4800
124	1010	30	46.265346	-54.2178	75	54.141419	27.8115
125	980	30	48.323469	-61.0781	80	57.572146	28.0348
126	982	30	51.434826	-71.4493	85	61.101037	28.1165
127	1008	35	35.000000	0.0000	0	0.000000	0.0000
128	968	35	35.274793	-0.7850	5	4.689415	6.2117
129	999	35	34.994994	0.0143	10	9.614985	3.8501
130	1018	35	36.057956	-3.0226	15	14.660418	2.2639
131	995	35	35.940703	-2.6876	20	17.998422	10.0079
132	1017	35	35.737463	-2.1069	25	22.338416	10.6463
133	997	35	38.430290	-9.8008	30	27.221950	9.2601
134	1040	35	38.185576	-9.1015	35	30.011535	14.2527
135	989	35	40.032355	-14.3780	40	33.010503	17.4737
136	1024	35	40.039062	-14.3972	45	35.172837	21.8381
137	963	35	42.201453	-20.5755	50	39.416521	21.1669
138	1030	35	44.822330	-28.0637	55	43.876721	20.2241
139	955	35	45.642931	-30.4082	60	45.255451	24.5742
140	1011	35	47.043521	-34.4099	65	47.399775	27.0772
141	1010	35	50.089108	-43.1117	70	53.023061	24.2527
142	1015	35	49.103448	-40.2955	75	53.951327	28.0649
143	1002	35	51.349301	-46.7122	80	57.785263	27.7685
144	1011	35	54.643916	-56.1254	85	60.419719	28.9179
145	1017	40	40.000000	0.0000	0	0.000000	0.0000
146	1009	40	39.980178	0.0495	5	4.657264	6.8547
147	1021	40	40.687561	-1.7188	10	9.514661	4.8534
148	995	40	40.953768	-2.3843	15	14.528260	3.1450
149	1004	40	41.440239	-3.6005	20	18.128031	9.3599
150	997	40	40.505516	-1.2637	25	22.489650	10.0414
151	998	40	42.831663	-7.0791	30	26.112513	12.9583
152	991	40	42.637739	-6.5942	35	29.931858	14.4804
153	1047	40	44.054441	-10.1360	40	33.971334	15.0716
154	978	40	45.840490	-14.6011	45	36.793417	18.2369
155	965	40	47.660103	-19.1502	50	40.693018	18.6140
156	955	40	47.986387	-19.9659	55	43.177524	21.4954
157	980	40	49.863265	-24.6581	60	48.212159	19.6464
158	1011	40	48.829871	-22.0746	65	49.044521	24.5469
159	980	40	52.439795	-31.0994	70	52.793443	24.5808
160	989	40	56.195146	-40.4878	75	57.717599	23.0432
161	986	40	58.634888	-46.5871	80	61.848501	22.6894
162	1010	40	59.219801	-48.0494	85	62.822332	26.0913
163	1034	45	45.000000	0.0000	0	0.000000	0.0000
164	1005	45	45.014925	-0.0330	5	4.726462	5.4708
165	1001	45	45.568431	-1.2631	10	9.302129	6.9787

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
166	1051	45	45.254043	-0.5644	15	14.686088	2.0928
167	1015	45	44.902463	0.2167	20	19.027161	4.8642
168	974	45	46.740246	-3.8671	25	23.702145	5.1914
169	996	45	46.922690	-4.2725	30	28.287666	5.7078
170	973	45	48.598150	-7.9957	35	30.284587	13.4726
171	1013	45	48.538993	-7.8643	40	34.484970	13.7876
172	1026	45	49.437621	-9.8613	45	39.073446	13.1701
173	999	45	49.453453	-9.8964	50	39.517906	20.9642
174	992	45	53.352822	-18.5617	55	44.400759	19.2713
175	975	45	54.604102	-21.3423	60	49.161900	18.0635
176	978	45	55.836400	-24.0808	65	50.320466	22.5839
177	1000	45	57.245999	-27.2132	70	57.179288	18.3153
178	1001	45	59.786213	-32.8581	75	55.332470	26.2233
179	974	45	62.544147	-38.9869	80	63.959051	20.0511
180	1051	45	59.478591	-32.1745	85	62.630453	26.3171
181	969	50	50.000000	0.0000	0	0.000000	0.0000
182	965	50	50.118134	-0.2362	5	4.776172	4.4766
183	1029	50	50.192419	-0.3848	10	9.637394	3.6260
184	1025	50	49.935609	0.1288	15	14.750325	1.6645
185	1007	50	50.977159	-1.9542	20	19.226766	3.8661
186	1020	50	50.564705	-1.1293	25	24.390606	2.4375
187	1035	50	51.792270	-3.5844	30	27.985219	6.7159
188	959	50	50.987486	-1.9749	35	31.894396	8.8731
189	969	50	52.970072	-5.9400	40	35.332952	11.6676
190	975	50	54.922051	-9.8440	45	39.845776	11.4539
191	1008	50	55.356150	-10.7122	50	41.875794	16.2484
192	992	50	52.441532	-4.8830	55	41.803328	23.9939
193	1023	50	55.288367	-10.5767	60	47.454293	20.9095
194	1013	50	59.481737	-18.9634	65	51.232757	21.1804
195	1026	50	60.980506	-21.9609	70	56.023738	19.9661
196	1008	50	60.164682	-20.3292	75	55.688938	25.7481
197	1017	50	65.410029	-30.8199	80	62.358202	22.0522
198	1014	50	67.196252	-34.3924	85	66.515917	21.7460
199	1012	55	55.000000	0.0000	0	0.000000	0.0000
200	1034	55	54.712765	0.5223	5	4.908435	1.8313
201	982	55	55.148676	-0.2702	10	9.553236	4.4676
202	997	55	55.244734	-0.4448	15	14.893339	0.7111
203	983	55	55.505595	-0.9192	20	19.438181	2.8091
204	997	55	56.267803	-2.3050	25	24.878853	0.4846
205	1013	55	56.461994	-2.6580	30	28.689385	4.3687
206	1033	55	57.509196	-4.5620	35	31.882509	8.9071
207	993	55	58.002014	-5.4581	40	35.232894	11.9177
208	966	55	57.104554	-3.8264	45	39.689840	11.8003
209	1007	55	60.659384	-10.2897	50	44.121828	11.7564
210	1000	55	60.160999	-9.3835	55	45.265949	17.6982
211	978	55	60.683026	-10.3326	60	47.802688	20.3289
212	984	55	63.873983	-16.1344	65	53.717531	17.3577
213	1039	55	62.681424	-13.9661	70	55.030466	21.3850
214	1022	55	66.243639	-20.4429	75	59.662177	20.4504
215	1053	55	64.653371	-17.5515	80	60.685458	24.1432
216	990	55	71.366666	-29.7576	85	67.359335	20.7537
217	992	60	60.000000	0.0000	0	0.000000	0.0000
218	995	60	59.948743	0.0854	5	4.711225	5.7755
219	993	60	60.804632	-1.3409	10	9.978714	0.2128
220	997	60	60.785356	-1.3089	15	15.042245	-0.2816
221	980	60	61.310204	-2.1836	20	19.365251	3.1738
222	1007	60	60.589870	-0.9830	25	23.592542	5.6299

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
223	972	60	62.211934	-3.6865	30	29.028739	3.2376
224	1008	60	62.028769	-3.3812	35	31.114553	11.1013
225	984	60	62.027601	-3.3959	40	36.580986	8.5475
226	1014	60	65.035502	-8.3924	45	39.254464	12.7678
227	1010	60	63.202970	-5.3381	50	41.807234	16.3855
228	1013	60	63.488647	-5.8143	55	45.761769	16.7968
229	1040	60	60.697115	-1.1617	60	49.052422	18.2459
230	1005	60	64.525373	-7.5422	65	52.997788	18.4649
231	977	60	66.808597	-11.3475	70	54.297877	22.4316
232	954	60	72.053459	-20.0890	75	62.745394	16.3394
233	979	60	70.961184	-18.2685	80	62.103965	22.3700
234	1008	60	75.744047	-26.2400	85	66.024235	22.3244
235	993	65	65.000000	0.0000	0	0.000000	0.0000
236	1012	65	65.087944	-0.1352	5	4.542715	9.1457
237	1021	65	65.645445	-0.9929	10	9.267576	7.3243
238	966	65	65.047619	-0.0732	15	14.832250	1.1183
239	1017	65	65.839724	-1.2918	20	20.557174	-2.7857
240	998	65	65.217434	-0.3344	25	24.939604	0.2416
241	1002	65	65.455089	-0.7000	30	28.686263	4.3791
242	1019	65	68.390578	-5.2162	35	32.361821	7.5377
243	942	65	67.676220	-4.1172	40	38.019128	4.9522
244	981	65	68.536187	-5.4402	45	39.479341	12.2681
245	1014	65	68.797830	-5.8427	50	44.977120	10.0458
246	986	65	68.901622	-6.0024	55	46.876417	14.7702
247	1020	65	69.105882	-6.3166	60	51.670985	13.8817
248	997	65	72.290872	-11.2166	65	52.884846	18.6387
249	985	65	69.128934	-6.3521	70	57.349011	18.0728
250	1021	65	72.994123	-12.2985	75	61.260773	18.3190
251	984	65	73.252032	-12.6954	80	64.362383	19.5470
252	987	65	79.526849	-22.3489	85	67.387604	20.7204
253	1009	70	70.000000	0.0000	0	0.000000	0.0000
254	989	70	70.036400	-0.0519	5	4.660405	6.7919
255	992	70	70.604838	-0.8640	10	10.166187	-1.6617
256	1006	70	70.365805	-0.5225	15	15.049087	-0.3272
257	976	70	71.440573	-2.0579	20	19.590595	2.0470
258	1011	70	70.437190	-0.6244	25	24.799051	0.8038
259	1008	70	70.991071	-1.4157	30	28.909525	3.6349
260	1037	70	70.999035	-1.4271	35	33.423327	4.5048
261	1022	70	74.445205	-6.3502	40	39.273052	1.8174
262	995	70	73.556783	-5.0810	45	40.817176	9.2951
263	1009	70	73.778989	-5.3984	50	44.850697	10.2986
264	970	70	74.821649	-6.8880	55	47.916187	12.8797
265	1006	70	76.242544	-8.9179	60	52.733715	12.1105
266	1012	70	76.976284	-9.9660	65	54.698648	15.8482
267	1028	70	75.202334	-7.4318	70	58.831729	15.9546
268	1031	70	81.325897	-16.1798	75	62.900850	16.1322
269	972	70	79.509259	-13.5846	80	63.127557	21.0906
270	1029	70	79.370262	-13.3860	85	65.190158	23.3057
271	993	75	75.000000	0.0000	0	0.000000	0.0000
272	1018	75	75.219056	-0.2920	5	4.614574	7.7085
273	965	75	75.327461	-0.4365	10	9.485441	5.1456
274	993	75	74.680765	0.4257	15	15.148121	-0.9874
275	1017	75	74.470009	0.7066	20	20.000429	-0.0020
276	989	75	76.127401	-1.5031	25	25.663818	-2.6552
277	971	75	76.141091	-1.5213	30	30.573854	-1.9127
278	1076	75	76.465613	-1.9540	35	32.325636	7.6411
279	1006	75	78.517892	-4.6904	40	38.994313	2.5142

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
280	991	75	80.686175	-7.5815	45	42.967729	4.5162
281	1021	75	78.469147	-4.6255	50	46.557853	6.8843
282	1012	75	78.607787	-4.8102	55	47.625302	13.4085
283	993	75	79.680765	-6.2409	60	52.985693	11.6905
284	980	75	84.789795	-13.0530	65	56.912983	12.4416
285	997	75	82.032096	-9.3760	70	61.203238	12.5668
286	1005	75	81.961194	-9.2815	75	62.309802	16.9202
287	967	75	82.735263	-10.3136	80	65.137363	18.5783
288	962	75	82.825363	-10.4337	85	66.505308	21.7585
289	993	80	80.000000	0.0000	0	0.000000	0.0000
290	982	80	80.084521	-0.1055	5	4.633130	7.3374
291	978	80	80.733128	-0.9163	10	8.941724	10.5828
292	980	80	80.697959	-0.8724	15	14.667949	2.2137
293	1006	80	80.361829	-0.4522	20	19.595366	2.0232
294	1010	80	81.727722	-2.1595	25	24.250228	2.9991
295	1001	80	80.821178	-1.0264	30	28.215299	5.9490
296	957	80	79.898641	0.1267	35	33.661172	3.8252
297	982	80	81.403258	-1.7539	40	38.475860	3.8104
298	1011	80	82.743818	-3.4296	45	43.661894	2.9736
299	990	80	79.811111	0.2361	50	47.137898	5.7242
300	966	80	85.817805	-7.2722	55	51.447563	6.4590
301	1009	80	81.918731	-2.3983	60	51.836619	13.6056
302	1038	80	86.850674	-8.5633	65	56.204629	13.5313
303	1037	80	85.675024	-7.0937	70	60.328184	13.8169
304	1008	80	85.421626	-6.7770	75	61.877718	17.4963
305	991	80	89.312815	-11.6409	80	67.884699	15.1441
306	1012	80	88.062252	-10.0777	85	67.889035	20.1306
307	984	85	85.000000	0.0000	0	0.000000	0.0000
308	988	85	85.021255	-0.0249	5	4.629599	7.4080
309	1023	85	84.835777	0.1932	10	9.700628	2.9937
310	1011	85	85.174085	-0.2047	15	14.195989	5.3601
311	999	85	85.711711	-0.8372	20	19.523150	2.3842
312	1024	85	85.895507	-1.0535	25	24.240525	3.0379
313	986	85	85.302231	-0.3555	30	29.366439	2.1119
314	962	85	88.073804	-3.6162	35	33.517322	4.2363
315	984	85	86.084349	-1.2756	40	38.454953	3.8626
316	1026	85	87.410331	-2.8356	45	43.435540	3.4766
317	1002	85	86.974051	-2.3223	50	48.258269	3.4834
318	949	85	88.825079	-4.5000	55	50.640777	7.9258
319	1003	85	89.488534	-5.2805	60	51.970038	13.3832
320	1038	85	89.951830	-5.8256	65	58.011202	10.7520
321	981	85	88.321100	-3.9071	70	58.920856	15.8273
322	962	85	92.075883	-8.3245	75	62.550987	16.5987
323	997	85	91.845536	-8.0534	80	63.867460	20.1657
324	981	85	92.396534	-8.7018	85	66.799446	21.4124
325	988	90	90.000000	0.0000	0	0.000000	0.0000
326	990	90	90.286868	-0.3187	5	4.518096	9.6381
327	984	90	90.023373	-0.0259	10	9.269703	7.3030
328	1022	90	89.735812	0.2936	15	14.410815	3.9279
329	972	90	90.483539	-0.5372	20	18.683684	6.5815
330	1017	90	90.095378	-0.1059	25	24.632812	1.4687
331	1013	90	90.097729	-0.1085	30	29.925523	0.2483
332	998	90	91.067134	-1.1857	35	34.031558	2.7669
333	1008	90	89.588293	0.4574	40	40.396621	-0.9915
334	1025	90	91.744390	-1.9381	45	42.013891	6.6358
335	1006	90	91.078528	-1.1982	50	46.833620	6.3327
336	1018	90	94.293713	-4.7707	55	51.530262	6.3086

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
337	1000	90	93.903999	-4.3376	60	53.889767	10.1838
338	979	90	95.478038	-6.0866	65	58.484528	10.0238
339	1037	90	93.400192	-3.7779	70	59.363557	15.1949
340	1012	90	91.894268	-2.1046	75	63.552184	15.2638
341	1032	90	95.624031	-6.2488	80	64.226141	19.7174
342	992	90	95.755040	-6.3944	85	65.692565	22.7147
343	1022	95	95.000000	0.0000	0	0.000000	0.0000
344	978	95	95.056237	-0.0591	5	4.712181	5.7564
345	1012	95	95.226284	-0.2381	10	9.414685	5.8531
346	1009	95	95.236868	-0.2492	15	15.148485	-0.9898
347	984	95	95.967479	-1.0183	20	20.186442	-0.9321
348	1022	95	95.027397	-0.0288	25	23.868216	4.5271
349	990	95	94.852525	0.1952	30	29.589071	1.3698
350	1032	95	95.078488	-0.0825	35	34.884438	0.3302
351	963	95	97.961578	-3.1173	40	39.951239	0.1219
352	961	95	97.278876	-2.3987	45	43.895891	2.4536
353	999	95	95.443443	-0.4667	50	47.814905	4.3702
354	967	95	97.304033	-2.4252	55	52.153030	5.1763
355	984	95	96.332317	-1.4023	60	53.262821	11.2286
356	1005	95	101.228855	-6.5566	65	59.426303	8.5749
357	994	95	97.604627	-2.7416	70	61.067375	12.7609
358	989	95	99.821031	-5.0746	75	63.235049	15.6866
359	1016	95	102.457677	-7.8501	80	67.064855	16.1689
360	988	95	102.932186	-8.3496	85	68.019891	19.9766
361	1012	100	100.000000	0.0000	0	0.000000	0.0000
362	996	100	99.840361	0.1596	5	4.599072	8.0185
363	985	100	100.159390	-0.1593	10	9.883577	1.1642
364	995	100	100.278391	-0.2783	15	14.598450	2.6770
365	975	100	99.234871	0.7651	20	18.862883	5.6856
366	1007	100	99.193644	0.8064	25	23.445663	6.2173
367	943	100	100.527041	-0.5270	30	29.087377	3.0421
368	1001	100	100.067932	-0.0679	35	35.619171	-1.7689
369	1016	100	102.081692	-2.0816	40	40.322327	-0.8057
370	1014	100	99.958579	0.0414	45	43.166662	4.0740
371	979	100	101.184882	-1.1848	50	48.099246	3.8015
372	1034	100	103.527079	-3.5270	55	51.035832	7.2075
373	987	100	102.862208	-2.8621	60	55.455786	7.5736
374	1035	100	103.230917	-3.2308	65	58.609549	9.8315
375	981	100	106.592252	-6.5921	70	60.528447	13.5308
376	1019	100	106.185475	-6.1854	75	64.509149	13.9878
377	1027	100	106.852969	-6.8529	80	66.150391	17.3120
378	974	100	109.647843	-9.6477	85	68.873018	18.9729
379	1027	105	105.000000	0.0000	0	0.000000	0.0000
380	999	105	105.390390	-0.3717	5	4.807708	3.8458
381	1021	105	106.041136	-0.9915	10	9.685156	3.1484
382	997	105	105.806419	-0.7679	15	14.626669	2.4888
383	1039	105	105.629451	-0.5993	20	19.480250	2.5987
384	1023	105	106.115347	-1.0622	25	24.413920	2.3443
385	998	105	106.080160	-1.0286	30	29.428212	1.9060
386	1008	105	103.997023	0.9552	35	33.432336	4.4791
387	976	105	106.436475	-1.3680	40	38.081803	4.7955
388	994	105	108.308853	-3.1512	45	43.802566	2.6610
389	1027	105	105.703992	-0.6704	50	48.700301	2.5994
390	994	105	107.202213	-2.0973	55	51.639918	6.1093
391	982	105	105.117107	-0.1114	60	52.917007	11.8050
392	982	105	106.442973	-1.3742	65	58.368180	10.2028
393	988	105	106.948380	-1.8555	70	61.430238	12.2426

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN ISD	CALC ISD	% ERROR
394	985	105	110.032487	-4.7927	75	64.938158	13.4158
395	1008	105	108.512896	-3.3455	80	65.790208	17.7622
396	1040	105	109.130769	-3.9340	85	69.275101	18.4999
397	1043	110	110.000000	0.0000	0	0.000000	0.0000
398	1006	110	110.177932	-0.1617	5	4.581298	8.3741
399	1003	110	110.315054	-0.2863	10	9.483709	5.1630
400	981	110	110.090723	-0.0823	15	15.030210	-0.2013
401	1011	110	110.561819	-0.5106	20	20.159785	-0.7988
402	970	110	112.101030	-1.9100	25	24.677241	1.2910
403	997	110	110.387161	-0.3519	30	29.143128	2.8562
404	971	110	109.578784	0.3829	35	32.996702	5.7237
405	981	110	110.516819	-0.4697	40	39.244434	1.8889
406	1019	110	111.539744	-1.3997	45	43.234606	3.9231
407	1028	110	112.439688	-2.2178	50	47.870844	4.2583
408	1015	110	110.948768	-0.8625	55	54.940827	0.1076
409	985	110	110.940101	-0.8545	60	55.994932	6.6751
410	1006	110	112.669980	-2.4271	65	59.581145	8.3367
411	1051	110	111.236917	-1.1243	70	62.748402	10.3594
412	993	110	111.335347	-1.2139	75	64.561559	13.9179
413	992	110	115.993951	-5.4490	80	70.182378	12.2720
414	1025	110	113.923902	-3.5671	85	71.528459	15.8489
415	981	115	115.000000	0.0000	0	0.000000	0.0000
416	999	115	114.906906	0.0809	5	4.788361	4.2327
417	1017	115	115.088495	-0.0768	10	9.430546	5.6945
418	1017	115	115.403146	-0.3504	15	14.225366	5.1642
419	991	115	116.402623	-1.2196	20	19.960478	0.1976
420	996	115	114.427710	0.4977	25	23.797090	4.8116
421	1012	115	115.872529	-0.7586	30	30.773681	-2.5789
422	977	115	114.416581	0.5073	35	34.941953	0.1659
423	1016	115	115.863188	-0.7505	40	39.384978	1.5376
424	959	115	113.207507	1.5587	45	45.395431	-0.8787
425	1056	115	117.512310	-2.1845	50	50.045817	-0.0915
426	966	115	113.828157	1.0190	55	51.144208	7.0105
427	1028	115	113.720817	1.1123	60	56.969618	5.0506
428	1001	115	117.051948	-1.7842	65	58.671861	9.7356
429	1015	115	117.628571	-2.2856	70	62.338177	10.9455
430	1035	115	118.504347	-3.0471	75	64.288810	14.2816
431	1008	115	118.129960	-2.7216	80	68.557082	14.3036
432	1002	115	120.257485	-4.5717	85	68.749495	19.1183
433	987	120	120.000000	0.0000	0	0.000000	0.0000
434	990	120	120.202020	-0.1683	5	4.478196	10.4361
435	964	120	120.814315	-0.6785	10	9.867862	1.3214
436	1021	120	121.168462	-0.9737	15	14.219397	5.2041
437	996	120	120.278112	-0.2317	20	18.813233	5.9338
438	1037	120	120.329797	-0.2747	25	25.190305	-0.7611
439	1002	120	121.770459	-1.4753	30	31.276301	-4.2542
440	1025	120	121.003902	-0.8365	35	34.345755	1.8692
441	1013	120	122.229022	-1.8574	40	38.291257	4.2718
442	1025	120	121.778536	-1.4821	45	44.766133	0.5197
443	992	120	121.978830	-1.6489	50	48.317219	3.3656
444	1002	120	122.171656	-1.8096	55	54.031280	1.7613
445	964	120	122.894190	-2.4117	60	57.126768	4.7887
446	993	120	117.309164	2.2423	65	59.860490	7.9070
447	1003	120	123.714855	-3.0956	70	61.400622	12.2848
448	991	120	121.097880	-0.9148	75	61.473109	18.0358
449	991	120	125.480322	-4.5668	80	66.786782	16.5165
450	1002	120	116.454091	2.9549	85	67.299297	20.8244

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
451	979	125	125.000000	0.0000	0	0.000000	0.0000
452	989	125	125.046511	-0.0371	5	4.904968	1.9006
453	977	125	125.544524	-0.4355	10	9.468364	5.3164
454	1013	125	126.116485	-0.8931	15	14.820414	1.1972
455	985	125	125.789847	-0.6318	20	19.791691	1.0416
456	978	125	125.904907	-0.7238	25	24.336400	2.6544
457	966	125	126.223602	-0.9788	30	30.152424	-0.5080
458	995	125	124.957788	0.0338	35	35.878602	-2.5102
459	1027	125	125.509250	-0.4073	40	39.235415	1.9115
460	1006	125	129.861829	-3.8893	45	43.203830	3.9915
461	1004	125	126.075697	-0.8605	50	49.065792	1.8684
462	996	125	125.686746	-0.5493	55	51.775418	5.8629
463	1008	125	127.865079	-2.2920	60	58.203609	2.9940
464	976	125	125.031762	-0.0253	65	57.563343	11.4410
465	1011	125	125.149357	-0.1194	70	61.128305	12.6738
466	989	125	129.896865	-3.9174	75	67.429012	10.0947
467	1008	125	126.116071	-0.8928	80	65.855681	17.6804
468	929	125	127.092572	-1.6739	85	70.517118	17.0387
469	991	130	130.000000	0.0000	0	0.000000	0.0000
470	1016	130	130.147637	-0.1134	5	4.488415	10.2317
471	1002	130	130.711576	-0.5473	10	9.700812	2.9919
472	998	130	129.780561	0.1688	15	14.874903	0.8339
473	977	130	130.463664	-0.3565	20	19.704503	1.4775
474	1010	130	132.370297	-1.8232	25	24.404209	2.3832
475	1001	130	130.884115	-0.6800	30	29.451261	1.8291
476	1020	130	130.521568	-0.4011	35	34.975004	0.0714
477	951	130	132.613038	-2.0099	40	38.379926	4.0502
478	1008	130	131.270833	-0.9775	45	43.450152	3.4441
479	1010	130	129.441584	0.4296	50	47.359422	5.2811
480	990	130	128.500000	1.1538	55	51.938533	5.5663
481	1038	130	131.722599	-1.3250	60	56.858270	5.2362
482	995	130	131.894578	-1.4573	65	60.022290	7.6581
483	968	130	130.060950	-0.0468	70	62.243554	11.0806
484	1014	130	131.193293	-0.9179	75	63.382263	15.4903
485	994	130	127.552313	1.8828	80	66.285157	17.1436
486	1005	130	130.163184	-0.1255	85	68.249886	19.7060
487	987	135	135.000000	0.0000	0	0.000000	0.0000
488	1007	135	135.248262	-0.1838	5	4.843472	3.1306
489	1019	135	135.432220	-0.3201	10	9.499667	5.0033
490	1008	135	135.160714	-0.1190	15	14.638783	2.4081
491	1036	135	136.110038	-0.8221	20	19.609265	1.9536
492	998	135	135.946893	-0.7013	25	23.780853	4.8766
493	995	135	135.378894	-0.2805	30	31.563297	-5.2109
494	960	135	135.490624	-0.3633	35	34.762596	0.6783
495	1015	135	138.461083	-2.5637	40	39.523722	1.1907
496	1037	135	138.558341	-2.6357	45	44.294162	1.5686
497	1024	135	133.787109	0.8984	50	48.205933	3.5881
498	998	135	136.030060	-0.7629	55	51.618684	6.1478
499	996	135	135.537148	-0.3978	60	55.792993	7.0116
500	1008	135	137.378968	-1.7621	65	58.739687	9.6312
501	1012	135	134.279644	0.5336	70	61.180441	12.5993
502	985	135	138.566497	-2.6417	75	65.509076	12.6545
503	1014	135	135.090729	-0.0671	80	68.374714	14.5316
504	1018	135	135.956777	-0.7086	85	68.513220	19.3963
505	1035	140	140.000000	0.0000	0	0.000000	0.0000
506	952	140	140.032563	-0.0231	5	4.660277	6.7545
507	1030	140	140.659223	-0.4708	10	9.850163	1.4984

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
508	1012	140	139.211462	0.5633	15	14.478773	3.4749
509	995	140	138.874371	0.8040	20	19.854691	0.7266
510	1019	140	140.403336	-0.2880	25	24.982536	0.0699
511	1021	140	141.568070	-1.1199	30	30.229525	-0.7650
512	973	140	143.900308	-2.7858	35	33.969082	2.9455
513	986	140	141.698782	-1.2133	40	38.953021	2.6174
514	1003	140	142.560319	-1.8287	45	42.964534	4.5232
515	1014	140	138.303747	1.2186	50	47.618160	4.7636
516	960	140	142.673958	-1.9099	55	50.645419	8.0992
517	945	140	138.079365	1.3719	60	55.163039	8.0616
518	1030	140	139.662135	0.2413	65	59.609114	8.2937
519	1001	140	138.670329	0.9498	70	60.732128	13.2398
520	1003	140	139.296111	0.5028	75	65.918465	12.1087
521	1007	140	137.751737	1.6059	80	68.125951	14.8426
522	1008	140	139.921626	0.0560	85	68.511678	19.3981
523	1023	145	145.000000	0.0000	0	0.000000	0.0000
524	1010	145	144.950495	0.0341	5	4.660903	6.7819
525	1005	145	145.303482	-0.2092	10	9.735680	2.6432
526	1024	145	145.488281	-0.3367	15	14.927257	0.4850
527	979	145	145.570990	-0.3937	20	19.924603	0.3769
528	1034	145	144.263056	0.5083	25	24.447871	2.2085
529	1001	145	145.158841	-0.1094	30	30.077064	-0.2568
530	975	145	145.209230	-0.1442	35	34.301889	1.9946
531	1030	145	144.967961	0.0221	40	38.770285	3.0743
532	1018	145	147.350687	-1.6211	45	44.028237	2.1595
533	1003	145	143.784646	0.8381	50	49.009383	1.9812
534	1035	145	147.195169	-1.5138	55	50.273705	8.5932
535	989	145	147.888776	-1.9922	60	55.303230	7.8280
536	1058	145	148.224952	-2.2240	65	57.239333	11.9395
537	994	145	143.111670	1.3023	70	62.700607	10.4277
538	1010	145	143.662376	0.9225	75	65.787370	12.2835
539	966	145	142.004140	2.0661	80	65.537539	18.0781
540	988	145	140.965587	2.7824	85	67.406108	20.6987
541	968	150	150.000000	0.0000	0	0.000000	0.0000
542	1034	150	150.299806	-0.1997	5	4.796662	4.0668
543	996	150	149.486947	0.3420	10	9.339980	6.6002
544	1042	150	149.803262	0.1312	15	15.407589	-2.7172
545	1030	150	150.858252	-0.5721	20	19.347154	3.2642
546	1018	150	150.690569	-0.4603	25	25.233457	-0.9337
547	1016	150	151.680118	-1.1199	30	29.192567	2.6914
548	1013	150	151.659427	-1.1062	35	34.416072	1.6684
549	1037	150	150.343297	-0.2287	40	39.581425	1.0464
550	997	150	149.961885	0.0254	45	44.400929	1.3313
551	1004	150	152.095617	-1.3970	50	45.844800	8.3104
552	974	150	152.697125	-1.7980	55	52.038306	5.3849
553	1016	150	150.247047	-0.1646	60	52.996728	11.6722
554	1046	150	151.278202	-0.8521	65	58.391241	10.1674
555	1022	150	147.375733	1.7495	70	61.766724	11.7618
556	1023	150	147.937438	1.3751	75	63.942582	14.7432
557	1026	150	148.368421	1.0877	80	65.986848	17.5165
558	1038	150	150.546242	-0.3641	85	68.963391	18.8666
559	1011	155	155.000000	0.0000	0	0.000000	0.0000
560	968	155	154.909090	0.0587	5	4.641242	7.1752
561	980	155	155.320408	-0.2066	10	9.497026	5.0297
562	991	155	155.202825	-0.1308	15	14.869885	0.8674
563	956	155	154.730125	0.1741	20	19.478825	2.6059
564	1017	155	155.274336	-0.1769	25	24.134477	3.4621

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
565	978	155	155.575664	-0.3713	30	29.839576	0.5348
566	1043	155	156.697027	-1.0948	35	35.146178	-0.4176
567	1019	155	154.025515	0.6287	40	38.111915	4.7203
568	964	155	156.157676	-0.7468	45	44.049346	2.1125
569	1005	155	154.249751	0.4840	50	46.287495	7.4250
570	993	155	155.941591	-0.6073	55	50.037009	9.0236
571	959	155	159.626694	-2.9849	60	53.089856	11.5169
572	1008	155	152.625992	1.5316	65	57.202110	11.9968
573	995	155	155.046231	-0.0297	70	57.429038	17.9585
574	994	155	154.299798	0.4517	75	62.667466	16.4434
575	997	155	153.976930	0.6601	80	65.944289	17.5697
576	986	155	146.595334	5.4224	85	68.826551	19.0276
577	1002	160	160.000000	0.0000	0	0.000000	0.0000
578	971	160	159.929969	0.0437	5	4.594177	8.1165
579	1030	160	160.078640	-0.0491	10	9.728325	2.7168
580	1005	160	159.865671	0.0840	15	14.657955	2.2803
581	974	160	160.152977	-0.0955	20	18.759754	6.2012
582	1028	160	160.452334	-0.2826	25	24.394854	2.4206
583	1009	160	159.745292	0.1592	30	29.071369	3.0955
584	1010	160	160.546534	-0.3415	35	35.025468	-0.0727
585	980	160	162.724489	-1.7027	40	40.169353	-0.4233
586	1006	160	160.404572	-0.2528	45	45.431272	-0.9583
587	985	160	162.942131	-1.8387	50	44.810758	10.3785
588	1030	160	159.533009	0.2919	55	49.796950	9.4601
589	1028	160	156.149805	2.4064	60	54.202378	9.6627
590	1016	160	157.265748	1.7089	65	58.913631	9.3636
591	979	160	159.049029	0.5944	70	58.945261	15.7925
592	976	160	157.732581	1.4171	75	63.814615	14.9139
593	1031	160	160.125121	-0.0781	80	65.910022	17.6125
594	960	160	157.303124	1.6856	85	67.532459	20.5500
595	1027	165	165.000000	0.0000	0	0.000000	0.0000
596	1031	165	165.071774	-0.0434	5	4.720233	5.5953
597	984	165	165.582317	-0.3528	10	9.334485	6.6552
598	983	165	165.710071	-0.4302	15	14.875054	0.8330
599	1004	165	165.765936	-0.4641	20	19.726041	1.3698
600	980	165	166.202040	-0.7284	25	25.041135	-0.1645
601	1022	165	164.967710	0.0196	30	30.076755	-0.2557
602	967	165	164.325749	0.4086	35	35.483157	-1.3804
603	984	165	163.735772	0.7862	40	37.908888	5.2278
604	974	165	166.472279	-0.8922	45	42.914131	4.6353
605	999	165	166.780780	-1.0791	50	48.343390	3.3132
606	1001	165	165.559440	-0.3390	55	49.366860	10.2420
607	1028	165	162.550583	1.4845	60	54.812352	8.6461
608	983	165	164.226856	0.4686	65	58.500496	9.9992
609	1011	165	160.525222	2.7120	70	60.764227	13.1939
610	986	165	163.307302	1.0259	75	62.671781	16.4376
611	1031	165	162.508244	1.5102	80	65.750926	17.8113
612	968	165	163.399793	0.9698	85	66.767701	21.4498
613	993	170	170.000000	0.0000	0	0.000000	0.0000
614	968	170	170.094008	-0.0552	5	4.540127	9.1975
615	992	170	170.134072	-0.0787	10	9.853485	1.4652
616	962	170	169.749480	0.1474	15	14.131267	5.7915
617	1013	170	169.737413	0.1544	20	19.678351	1.6082
618	1004	170	170.501992	-0.2952	25	24.071657	3.7134
619	1005	170	172.686567	-1.5802	30	29.553016	1.4900
620	981	170	170.688073	-0.4047	35	33.631961	3.9087
621	994	170	170.014084	-0.0082	40	38.433185	3.9171

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
622	1022	170	168.012720	1.1690	45	43.836363	2.5858
623	1007	170	168.765640	0.7261	50	47.585069	4.8298
624	1025	170	166.636097	1.9787	55	49.310177	10.3451
625	990	170	168.026262	1.1610	60	52.409269	12.6512
626	1008	170	165.147817	2.8542	65	56.846132	12.5444
627	1006	170	169.167992	0.4894	70	60.101180	14.1411
628	1046	170	165.641491	2.5638	75	61.764428	17.6474
629	1029	170	168.143828	1.0919	80	63.371173	20.7860
630	1011	170	166.136498	2.2727	85	67.726264	20.3221
631	938	175	175.000000	0.0000	0	0.000000	0.0000
632	1030	175	175.113592	-0.0648	5	4.745802	5.0840
633	1025	175	175.725853	-0.4147	10	9.770521	2.2948
634	937	175	174.610458	0.2226	15	13.659672	8.9353
635	990	175	176.062626	-0.6072	20	20.071465	-0.3572
636	991	175	175.480322	-0.2744	25	23.532562	5.8697
637	1018	175	175.465618	-0.2659	30	30.756435	-2.5214
638	1017	175	172.708947	1.3092	35	35.622227	-1.7777
639	1003	175	174.570289	0.2455	40	38.784138	3.0397
640	1023	175	176.097751	-0.6272	45	41.142091	8.5731
641	1006	175	174.645129	0.2028	50	45.808343	8.3833
642	992	175	171.158266	2.1953	55	48.418523	11.9663
643	1038	175	170.975915	2.2994	60	53.894057	10.1766
644	995	175	167.970854	4.0166	65	55.987928	13.8648
645	986	175	172.078093	1.6696	70	59.417417	15.1180
646	1011	175	168.412462	3.7643	75	62.564796	16.5802
647	955	175	171.073298	2.2439	80	61.455212	23.1810
648	990	175	167.452525	4.3128	85	69.189291	18.6008
649	1011	180	180.000000	0.0000	0	0.000000	0.0000
650	1023	180	179.867057	0.0739	5	4.679572	6.4085
651	992	180	180.426411	-0.2368	10	9.418872	5.8113
652	990	180	180.927272	-0.5150	15	14.784841	1.4344
653	992	180	181.262096	-0.7011	20	18.971169	5.1442
654	981	180	180.976554	-0.5424	25	25.347878	-1.3915
655	988	180	180.598178	-0.3322	30	29.854824	0.4839
656	1018	180	180.476424	-0.2646	35	35.432266	-1.2350
657	1047	180	179.682903	0.1762	40	37.273431	6.8164
658	978	180	179.598159	0.2232	45	41.951222	6.7750
659	1008	180	179.231150	0.4271	50	43.909420	12.1811
660	997	180	175.056168	2.7488	55	49.236000	10.4800
661	968	180	176.570247	1.9054	60	51.748612	13.7523
662	985	180	177.497461	1.3903	65	53.541001	17.6292
663	979	180	173.998978	3.3339	70	58.069199	17.0440
664	1006	180	173.422465	3.6541	75	63.783414	14.9555
665	1008	180	177.129960	1.5945	80	63.005998	21.2425
666	958	180	175.916492	2.2686	85	63.620589	25.1523
667	988	185	185.000000	0.0000	0	0.000000	0.0000
668	979	185	185.224719	-0.1214	5	4.766168	4.6767
669	951	185	185.674027	-0.3643	10	9.557860	4.4214
670	1036	185	185.872586	-0.4715	15	13.919559	7.2030
671	1018	185	185.810412	-0.4380	20	19.751719	1.2414
672	1007	185	184.829195	0.0924	25	25.551274	-2.2050
673	972	185	185.053497	-0.0288	30	30.131150	-0.4371
674	1005	185	185.956218	-0.5167	35	32.947779	5.8635
675	995	185	185.608040	-0.3286	40	35.789165	10.5271
676	1032	185	182.487403	1.3582	45	40.313007	10.4155
677	1003	185	182.469591	1.3678	50	44.180697	11.6386
678	977	185	182.555783	1.3212	55	44.659793	18.8004

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
679	1035	185	181.978743	1.6331	60	49.519820	17.4670
680	991	185	178.915237	3.2891	65	55.629191	14.4166
681	998	185	181.216632	2.0452	70	57.378798	18.0303
682	980	185	177.484693	4.0623	75	60.894113	18.8078
683	955	185	178.811518	3.3452	80	63.598651	20.5017
684	1031	185	172.427740	6.7958	85	65.975108	22.3822
685	1029	190	190.000000	0.0000	0	0.000000	0.0000
686	999	190	190.157157	-0.0826	5	4.583929	8.3214
687	992	190	190.296370	-0.1559	10	9.708528	2.9148
688	995	190	190.592964	-0.3120	15	14.980955	0.1270
689	988	190	189.724696	0.1449	20	19.364766	3.1762
690	1005	190	191.295522	-0.6817	25	23.733064	5.0678
691	1012	190	189.504940	0.2606	30	28.752899	4.1570
692	998	190	189.644288	0.1873	35	31.999231	8.5736
693	1011	190	187.849653	1.1318	40	35.769053	10.5773
694	985	190	190.718781	-0.3782	45	41.180489	8.4878
695	1010	190	185.992079	2.1094	50	45.222213	9.5556
696	993	190	186.712990	1.7300	55	45.692118	16.9234
697	950	190	186.278947	1.9584	60	51.018305	14.9695
698	984	190	186.660569	1.7576	65	51.964565	20.0545
699	1021	190	183.475024	3.4342	70	55.790616	20.2991
700	992	190	184.116935	3.0963	75	58.907601	21.4566
701	1039	190	183.852743	3.2354	80	61.183734	23.5203
702	978	190	180.763803	4.8611	85	65.656909	22.7566
703	1020	195	195.000000	0.0000	0	0.000000	0.0000
704	1027	195	195.192794	-0.0988	5	4.580027	8.3994
705	995	195	195.523618	-0.2684	10	10.229569	-2.2956
706	1026	195	195.680311	-0.3488	15	14.758263	1.6116
707	981	195	195.799184	-0.4097	20	19.787793	1.0610
708	1002	195	196.278443	-0.6555	25	23.514855	5.9406
709	979	195	194.063329	0.4803	30	27.839160	7.2028
710	1009	195	195.880079	-0.4512	35	32.961356	5.8247
711	1027	195	195.516066	-0.2646	40	35.244268	11.8893
712	1041	195	192.301633	1.3837	45	38.796997	13.7845
713	981	195	192.641182	1.2097	50	42.519189	14.9616
714	998	195	191.386773	1.8529	55	47.581244	13.4886
715	1005	195	190.571144	2.2712	60	49.466559	17.5558
716	1025	195	190.635121	2.2384	65	53.835759	17.1757
717	1005	195	185.992039	4.6195	70	56.831881	18.8116
718	987	195	185.747720	4.7448	75	61.234182	18.3545
719	1001	195	184.371628	5.4505	80	62.475034	21.9062
720	980	195	180.394897	7.4898	85	67.011322	21.1631
721	997	200	200.000000	0.0000	0	0.000000	0.0000
722	984	200	200.032520	-0.0162	5	5.000619	-0.0123
723	1041	200	200.153698	-0.0767	10	9.539369	4.6063
724	984	200	200.704268	-0.3520	15	15.210060	-1.4003
725	1018	200	200.741650	-0.3707	20	20.843169	-4.2158
726	984	200	199.768292	0.1158	25	24.470204	2.1192
727	1016	200	200.448818	-0.2243	30	28.207320	5.9756
728	958	200	199.624217	0.1879	35	31.574773	9.7864
729	965	200	201.033160	-0.5165	40	34.835255	12.9118
730	997	200	197.422266	1.2888	45	39.762416	11.6391
731	984	200	195.043699	2.4782	50	42.799958	14.4001
732	1038	200	194.767822	2.6161	55	46.742254	15.0141
733	943	200	195.607635	2.1962	60	45.542529	24.0958
734	1016	200	192.859251	3.5704	65	51.874092	20.1937
735	991	200	193.388496	3.3058	70	54.355466	22.3493

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
736	984	200	191.256097	4.3720	75	57.754256	22.9943
737	1032	200	190.082364	4.9588	80	58.538323	26.8271
738	1017	200	190.757128	4.6215	85	61.308270	27.8726
739	989	205	205.000000	0.0000	0	0.000000	0.0000
740	968	205	205.081611	-0.0397	5	4.696890	6.0622
741	1035	205	205.007729	-0.0037	10	9.380323	6.1967
742	999	205	205.313313	-0.1527	15	14.108123	5.9458
743	983	205	204.616480	0.1871	20	19.365349	3.1732
744	986	205	204.746450	0.1236	25	23.265379	6.9385
745	1024	205	204.706054	0.1434	30	27.799780	7.3341
746	953	205	205.216159	-0.1053	35	29.865730	14.6694
747	975	205	203.912820	0.5303	40	34.446515	13.8837
748	1003	205	202.623130	1.1594	45	38.408324	14.6482
749	1006	205	200.645129	2.1244	50	42.241527	15.5169
750	984	205	200.721544	2.0871	55	44.061664	19.8879
751	1008	205	196.898809	3.9518	60	48.687641	18.8539
752	1017	205	200.898721	2.0007	65	48.381449	25.5670
753	963	205	194.611630	5.0675	70	54.701561	21.8549
754	1004	205	196.640438	4.0779	75	54.411281	27.4517
755	996	205	190.078313	7.2789	80	62.891547	21.3855
756	1037	205	191.835101	6.4219	85	61.415831	27.7461
757	1022	210	210.000000	0.0000	0	0.000000	0.0000
758	1014	210	210.179487	-0.0854	5	4.716379	5.6724
759	1025	210	209.722926	0.1319	10	9.703800	2.9620
760	1048	210	210.564885	-0.2689	15	14.639958	2.4003
761	983	210	210.225839	-0.1074	20	19.219653	3.9017
762	990	210	210.169696	-0.0807	25	23.185755	7.2569
763	989	210	208.850353	0.5475	30	27.352035	8.8266
764	989	210	208.755308	0.5928	35	30.714252	12.2450
765	1040	210	208.554807	0.6882	40	32.748276	18.1293
766	985	210	207.964467	0.9693	45	34.986317	22.2526
767	1005	210	204.582089	2.5799	50	40.290512	19.4190
768	1045	210	206.314832	1.7548	55	42.106365	23.4430
769	1002	210	204.421157	2.6566	60	46.698734	22.1688
770	1001	210	204.069930	2.8239	65	50.004771	23.0696
771	973	210	200.491264	4.5280	70	54.010755	22.8418
772	1010	210	200.349504	4.5955	75	53.623675	28.5017
773	979	210	195.429009	6.9386	80	61.415227	23.2310
774	992	210	198.019153	5.7091	85	59.529798	29.9650
775	979	215	215.000000	0.0000	0	0.000000	0.0000
776	1053	215	215.092117	-0.0428	5	4.692888	6.1423
777	989	215	215.293225	-0.1363	10	9.691975	3.0802
778	1002	215	215.199600	-0.0927	15	14.412103	3.9193
779	1023	215	215.130987	-0.0608	20	17.780917	11.0954
780	989	215	214.253791	0.3471	25	22.475934	10.0963
781	1008	215	214.632936	0.1708	30	25.507449	14.9752
782	1031	215	212.301648	1.2551	35	29.601685	15.4237
783	1004	215	212.281872	1.2642	40	31.612771	20.9681
784	981	215	211.475025	1.6395	45	35.315528	21.5210
785	1021	215	210.181194	2.2413	50	39.632550	20.7349
786	1018	215	209.878192	2.3822	55	42.013589	23.6117
787	1041	215	206.191162	4.0971	60	46.651130	22.2481
788	967	215	209.646328	2.4901	65	45.675235	29.7304
789	1007	215	202.936444	5.6110	70	54.000407	22.8566
790	1016	215	203.905511	5.1602	75	54.764154	26.9811
791	1019	215	200.272816	6.8498	80	57.849818	27.6877
792	1031	215	203.588748	5.3075	85	58.380805	31.3167

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
793	1016	220	220.000000	0.0000	0	0.000000	0.0000
794	998	220	219.946893	0.0242	5	4.777972	4.4405
795	994	220	220.259557	-0.1179	10	9.119282	8.8071
796	983	220	220.284842	-0.1294	15	14.746202	1.6920
797	994	220	220.090543	-0.0411	20	17.862289	10.6885
798	1003	220	219.774675	0.1024	25	22.214691	11.1413
799	988	220	218.188259	0.8235	30	25.390586	15.3647
800	1004	220	217.425298	1.1703	35	28.319459	19.0873
801	1004	220	214.434262	2.5299	40	32.815005	17.9625
802	999	220	215.139139	2.2095	45	35.641469	20.7967
803	1004	220	214.627490	2.4420	50	36.361575	27.2768
804	971	220	211.075180	4.0567	55	42.244545	23.1917
805	986	220	213.600405	2.9089	60	41.646406	30.5893
806	1008	220	207.760912	5.5632	65	50.023327	23.0410
807	962	220	207.598752	5.6369	70	49.947645	28.6462
808	979	220	205.680286	6.5090	75	54.668689	27.1084
809	984	220	206.714430	6.0389	80	55.245969	30.9426
810	1014	220	203.173570	7.6484	85	59.333044	30.1964
811	972	225	225.000000	0.0000	0	0.000000	0.0000
812	987	225	225.090172	-0.0400	5	4.851870	2.9626
813	1013	225	224.867719	0.0588	10	9.747938	2.5206
814	1007	225	224.765640	0.1042	15	13.996695	6.6887
815	991	225	223.957618	0.4632	20	18.061859	9.6907
816	994	225	223.326961	0.7435	25	21.689353	13.2426
817	980	225	224.421428	0.2572	30	25.086636	16.3779
818	980	225	222.370408	1.1687	35	27.833347	20.4762
819	1030	225	220.215533	2.1265	40	31.064255	22.3394
820	1028	225	217.742217	3.2257	45	33.586596	25.3631
821	1030	225	217.793203	3.2031	50	35.942040	28.1159
822	1054	225	216.893738	3.6027	55	38.994948	29.1001
823	990	225	214.310101	4.7511	60	42.579252	29.0346
824	1028	225	215.698443	4.1340	65	43.360075	33.2921
825	973	225	212.894141	5.3804	70	49.033481	29.9522
826	1020	225	211.137254	6.1613	75	49.836584	33.5513
827	964	225	212.366182	5.6150	80	52.780388	34.0245
828	957	225	206.832810	8.0743	85	59.400171	30.1175
829	1028	230	230.000000	0.0000	0	0.000000	0.0000
830	1010	230	230.015841	-0.0068	5	4.737721	5.2455
831	1005	230	230.123383	-0.0538	10	9.850412	1.4959
832	1023	230	229.423264	0.2508	15	13.767149	8.2190
833	1016	230	229.658464	0.1485	20	17.248652	13.7567
834	995	230	227.724623	0.9893	25	20.316871	18.7325
835	1053	230	225.403608	1.9985	30	24.920440	16.9319
836	989	230	223.951466	2.6298	35	28.245843	19.2975
837	959	230	225.507820	1.9531	40	28.871393	27.8215
838	979	230	224.655771	2.3236	45	30.439673	32.3563
839	993	230	221.832829	3.5510	50	35.829071	28.3418
840	1019	230	219.046123	4.7626	55	39.221189	28.6888
841	1008	230	218.388888	5.0483	60	42.617417	28.9710
842	959	230	215.544316	6.2850	65	44.796550	31.0822
843	1017	230	212.939036	7.4178	70	50.073335	28.4667
844	1010	230	212.167326	7.7533	75	51.530154	31.2932
845	1002	230	212.622754	7.5553	80	54.081556	32.3981
846	1022	230	210.572407	8.4467	85	57.591796	32.2449
847	979	235	235.000000	0.0000	0	0.000000	0.0000
848	976	235	235.129098	-0.0549	5	4.647348	7.0530
849	993	235	234.998992	0.0004	10	9.420156	5.7985

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
850	977	235	233.825997	0.4996	15	12.998536	13.3431
851	989	235	232.930232	0.8808	20	16.587775	17.0611
852	972	235	231.613168	1.4412	25	19.770851	20.9166
853	1017	235	232.179941	1.2001	30	21.789121	27.3696
854	973	235	230.825282	1.7765	35	24.207789	30.8349
855	1023	235	228.768328	2.6517	40	27.891358	30.2716
856	1008	235	226.144841	3.7681	45	31.428596	30.1587
857	940	235	227.668085	3.1200	50	31.601055	36.7979
858	990	235	223.332323	4.9650	55	39.044577	29.0099
859	1006	235	221.798210	5.6178	60	40.271619	32.8806
860	1028	235	219.324902	6.6702	65	43.533127	33.0260
861	953	235	219.330535	6.6679	70	47.496361	32.1480
862	1029	235	217.357628	7.5074	75	49.569515	33.9073
863	998	235	216.154308	8.0194	80	51.820774	35.2240
864	982	235	211.955193	9.8063	85	58.395591	31.2993
865	950	240	240.000000	0.0000	0	0.000000	0.0000
866	1001	240	240.223776	-0.0931	5	4.736874	5.2625
867	1014	240	240.261341	-0.1088	10	8.837451	11.6255
868	1001	240	238.707292	0.5386	15	12.712260	15.2516
869	1020	240	238.019607	0.8252	20	15.305332	23.4733
870	989	240	235.967644	1.6801	25	18.431210	26.2752
871	1020	240	235.100980	2.0413	30	20.882605	30.3914
872	1027	240	233.129503	2.8627	35	24.591833	29.7376
873	1020	240	232.331372	3.1952	40	26.977187	32.5571
874	996	240	230.461847	3.9742	45	29.061565	35.4187
875	981	240	228.770642	4.6789	50	33.211405	33.5772
876	994	240	226.527162	5.6137	55	36.868512	32.9664
877	1015	240	226.130049	5.7791	60	39.578535	34.0357
878	976	240	223.526639	6.8639	65	41.531001	36.1062
879	935	240	222.347593	7.3552	70	45.087712	35.5890
880	1003	240	221.939182	7.5253	75	45.379256	39.4944
881	973	240	218.201438	9.0827	80	53.134069	33.5824
882	1028	240	216.830739	9.6539	85	53.426037	37.1459
883	1008	245	245.000000	0.0000	0	0.000000	0.0000
884	1015	245	244.830541	0.0692	5	4.560494	8.7902
885	998	245	244.867735	0.0540	10	7.846110	21.5389
886	983	245	243.126144	0.7648	15	11.703941	21.9738
887	1026	245	242.160818	1.1589	20	14.067531	29.6624
888	986	245	239.906693	2.0780	25	17.369608	30.5215
889	1009	245	238.566897	2.6258	30	20.406488	31.9784
890	995	245	236.665326	3.4019	35	24.348861	30.4318
891	984	245	235.693089	3.7987	40	26.169677	34.5758
892	1004	245	233.697211	4.6133	45	29.225291	35.0549
893	987	245	233.446808	4.7156	50	31.237917	37.5242
894	1039	245	228.040423	6.9223	55	38.481441	30.0337
895	974	245	228.822381	6.6031	60	37.336951	37.7717
896	1018	245	228.564833	6.7082	65	39.294193	39.5474
897	980	245	225.860204	7.8122	70	43.535268	37.8067
898	988	245	222.752024	9.0808	75	48.076395	35.8982
899	1004	245	221.619521	9.5431	80	50.657620	36.6780
900	988	245	221.310728	9.6691	85	52.618390	38.0960
901	998	250	250.000000	0.0000	0	0.000000	0.0000
902	998	250	249.713426	0.1146	5	3.950098	20.9980
903	997	250	248.356068	0.6576	10	7.157626	28.4237
904	1007	250	246.926514	1.2294	15	10.182175	32.1188
905	1000	250	245.922999	1.6308	20	12.883071	35.5847
906	970	250	243.928865	2.4285	25	15.617412	37.5304

CASE	GENERATED SAMPLES	GIVEN VI	CALC MEAN	% ERROR	GIVEN 1SD	CALC 1SD	% ERROR
907	993	250	241.354481	3.4582	30	20.267997	32.4400
908	988	250	240.415991	3.8336	35	21.197112	39.4369
909	1017	250	238.557522	4.5770	40	25.949469	35.1263
910	990	250	235.757575	5.6970	45	28.045142	37.6774
911	1004	250	234.195219	6.3220	50	32.341211	35.3176
912	975	250	233.895384	6.4418	55	33.370541	39.3263
913	1057	250	232.570482	6.9718	60	35.261915	41.2302
914	965	250	230.413471	7.8346	65	40.692861	37.3956
915	1014	250	225.073964	9.9704	70	46.271545	33.8978
916	1031	250	224.929194	10.0283	75	47.034507	37.2873
917	1010	250	224.500990	10.1996	80	48.323350	39.5958
918	975	250	220.824615	11.6702	85	55.473092	34.7375
919	1037	255	255.000000	0.0000	0	0.000000	0.0000
920	1010	255	253.489108	0.5925	5	2.846388	43.0723
921	1001	255	251.691308	1.2975	10	5.869579	41.3043
922	993	255	250.437059	1.7894	15	8.415646	43.8957
923	995	255	247.760804	2.8389	20	12.436522	37.8174
924	1039	255	246.307988	3.4087	25	15.002751	39.9890
925	994	255	244.578470	4.0868	30	18.246681	39.1777
926	1020	255	243.070588	4.6782	35	20.789071	40.6026
927	1020	255	241.221568	5.4033	40	22.762765	43.0931
928	1013	255	241.127344	5.4403	45	25.865852	42.5203
929	999	255	238.674674	6.4021	50	29.205470	41.5890
930	995	255	235.252261	7.7442	55	32.843170	40.2852
931	1001	255	235.144855	7.7863	60	33.658907	43.9018
932	994	255	231.557344	9.1932	65	40.111491	38.2900
933	1017	255	230.915437	9.4449	70	41.831365	40.2409
934	966	255	229.312629	10.0735	75	43.632684	41.8231
935	1012	255	228.499011	10.3925	80	45.738185	42.8273
936	1065	255	228.446009	10.4133	85	47.522524	44.0912

DISTRIB

program listing and output

prepared for

Dr. Daniel W. Scott III
CSCI 488

by

Daniel Paul Long
September 18, 1979

```
*PL/C SORMGIN=(2,80,1),TIME=(2,0)

*OPTIONS IN EFFECT*      PAGES=030,TIME=(002,000),ERRORS=(050,0501,SORMGIN=(002,080,001),LINECNT=060,BOUNDARY,
*OPTIONS IN EFFECT*      FLAGW,NOXREF,NOATR,NOLIST,UDEF,SOURCE,DUMP,NODUMPARRAY,NOM91,NOCOMMENTS,CHECK

     /*    DISTRIBUTION    */                                          PL/C-R6.6000  09/18/79  9:06 PAGE   1

    STMT LEVEL NEST BLOCK      SOURCE STATEMENT                                                    ID FIELD

                               /*    DISTRIBUTION    */

                               /*    DISTRIB SIMULATES THE USE OF THE SUBROUTINE DEVIAT IN THE PROGRAM      */
                               /*  VISORP. THIS PROGRAM PRODUCES A HISTOGRAPH OF THE OCCURRENCE OF TIME     */
                               /*  INTERVALS FOR A GIVEN VI AND SD.                                        */

       1                       DIST:PROC OPTIONS(MAIN);

       2    1         1          DCL X FIXED(9,9),(MEAN,DEV) FIXED(10,6),(VIERR,SDERR) INIT(0),((N,SUM,SUMSQ)
                                   INIT(0),V INIT(1),POINTR INIT(1023),LINES(0:127) INIT((128)0),VI,SD,ITER,
                                   J,I,VALUES(0:1023),B INIT(1),RNDSGN(32)) BIN FIXED(31),BYTE CHAR(2),
                                   HEX(0:15)
                                   CHAR(1) INIT('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E',
                                   'F');

                               /* USE THE TIME STRING AS RANDOM SEED X */
       3    1         1          GET STRING(TIME) EDIT(X)(F(9,9));

                               /* RNDSGN IS AN ARRAY USED TO SIMULATE A FOUR BYTE DATA BASE CONTAINING     */
                               /* RANDOM 1'S AND 0'S.                                                     */
       4    1         1          GET LIST(RNDSGN);

                               /* READ IN THE SAMPLE DATA BASE ONE VALUE AT A TIME AND CONVERT TO BASE 10  */
       5    1         1          DO I=0 TO 1023;
       6    1    1    1            GET LIST(BYTE);
       7    1    1    1            DO J=0 TO 15 WHILE (SUBSTR(BYTE,1,1)¬=HEX(J)); END;
       9    1    1    1            VALUES(I) = J * 16;
      10    1    1    1            DO J=0 TO 15 WHILE (SUBSTR(BYTE,2)¬=HEX(J)); END;
      12    1    1    1            VALUES(I) = VALUES(I) + J;
      13    1    1    1            END;

                               /* READ THE PARAMETER PAIR AND THE NUMBER OF ITERATIONS. */
      14    1         1          GET LIST(VI,SD,ITER);


      15    1         1          DO WHILE (M < ITER);

      16    1    1    1            IF X < 0.5 THEN DO;

      18    1    2    1              T = VALUES(POINTR) * 3 * SD / 256;
      19    1    2    1              T = VI + T * RNDSGN(B);

      20    1    2    1              IF T > 255 THEN T = 255;
      22    1    2    1              IF T < 0 THEN T = 0;
```

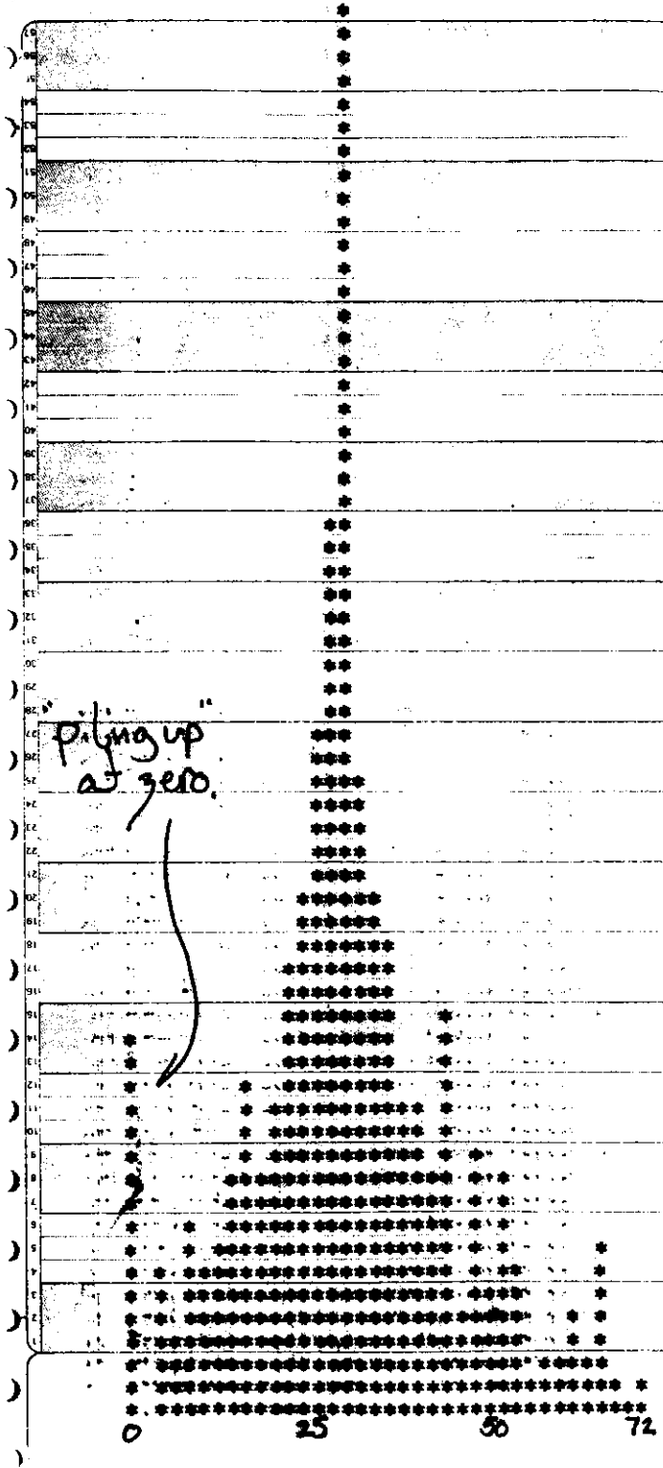
```
STMT LEVEL NEST BLOCK    SOURCE STATEMENT                                                    ID FIELD
                                 /* "ROLL" THE SIMULATED BITS OF RNDSGN TO THE LEFT A BIT AT A TIME.          */
   24     1    2    1             IF B = 32 THEN B = 1;
   26     1    2    1                       ELSE B = B + 1;

                                 /* RECORD STATISTICAL DATA FOR THIS ITERATION.                               */
   27     1    2    1               SUM = SUM + T;
   28     1    2    1               SUMSQ = SUMSQ + T**2;
   29     1    2    1               T = ROUND(T / 2 + 1,0) - 1;
   30     1    2    1               LINES(T) = LINES(T) + 1;

   31     1    2    1               M = M + 1;
   32     1    2    1               END;

   33     1    1    1             X = RAND(X);
                                 /* MAINTAIN THE POINTR OFFSET IN THE SAMPLE DATA BASE. */
   34     1    1    1             IF POINTR = 0 THEN POINTR = 1023;
   36     1    1    1                           ELSE POINTR = POINTR - 1;

   37     1    1    1             END;


                                 /* MEAN, DEV, VIERR, AND SDERR CAN BE FOUND IN THE POST EXECUTION DUMP.      */
   38     1         1           MEAN = FLOAT(SUM) / FLOAT(ITER);
   39     1         1           DEV = SQRT((SUMSQ - ITER * MEAN**2) / (ITER - 1));

   40     1         1           IF VI¬=0 THEN VIERR = (1 - MEAN / VI);
   42     1         1           IF SD¬=0 THEN SDERR = (1 - DEV / SD);

   44     1         1           PUT PAGE;
   45     1         1           ON ENDPAGE;

                                 /* PRINT THE HISTOGRAPH. GETS PRINTED UPSIDE-DOWN. */

                                 /* THIS LOOP DOES THE VERTICAL PART OF THE HISTOGRAPH. */
   46     2         2           DO I=1 BY 1 WHILE (V = 1);
   48     1    1    1             V = 0;

                                 /* THIS LOOP DOES THE HORIZONTAL PART OF THE HISTOGRAPH. */
   49     1    1    1             DO J=0 TO 127;
   50     1    2    1               IF LINES(J) >= I THEN DO;
   52     1    3    1                 PUT SKIP(0) EDIT(SUBSTR(REPEAT(' ',126),1,127-J)||'*')(COL(3),A);
                                 /* V IS USED AS A SWITCH. WHEN NO ASTERISKS ARE PRINTED IN A HORIZONTAL LINE,*/
                                 /* THE HISTOGRAPH IS THROUGH . */
   53     1    3    1                 V = 1;
   54     1    3    1                 END;
   55     1    2    1               END;

   56     1    1    1             PUT SKIP;
   57     1    1    1             END;

   58     1         1           ON ENDPAGE SYSTEM;
   59     1         1           PUT PAGE; PUT PAGE;

   61     1         1           END DIST;
```

(continued at bottom of next page)

VI = 30 μ = 31.039999 3.46% error

SD = 15 σ = 14.386814 4.09 % error

no. of iterations = 500

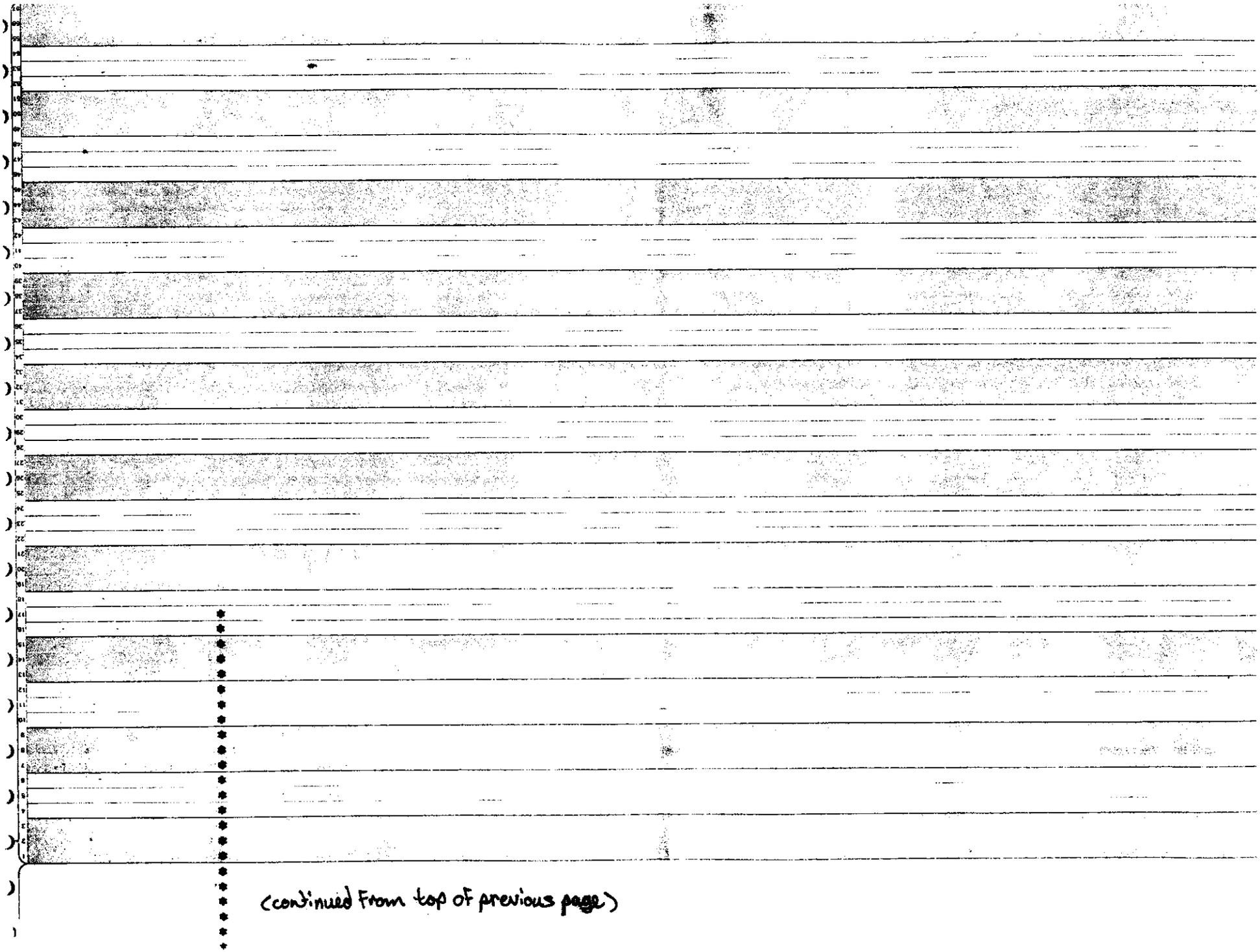


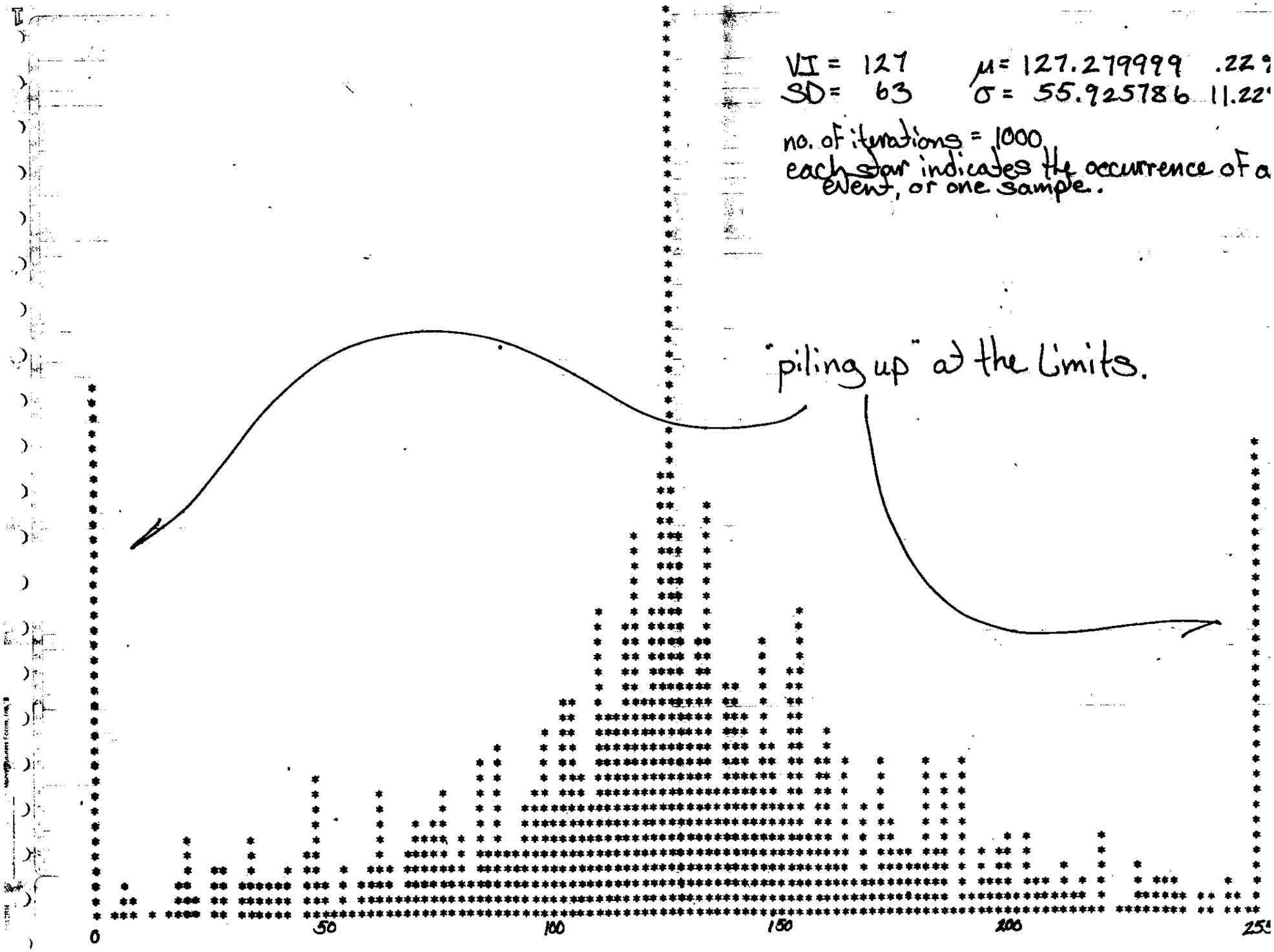
notice that no events (time intervals) occur past three standard deviations away from the mean, where one standard deviation is defined to be 15 and the mean is 30 (see above).

$1\sigma = 15 \rightarrow 3\sigma$'s = 45 $\rightarrow$ $\mu + 3\sigma$'s = 30 + 45 = 75

$\mu = 30$

this is because the scaled, static data base used by the DEVIAT / subroutine and this program contains only values within three standard deviations of the mean. Hence no values can be generated outside of three standard deviations of the mean using this implementation of this method.

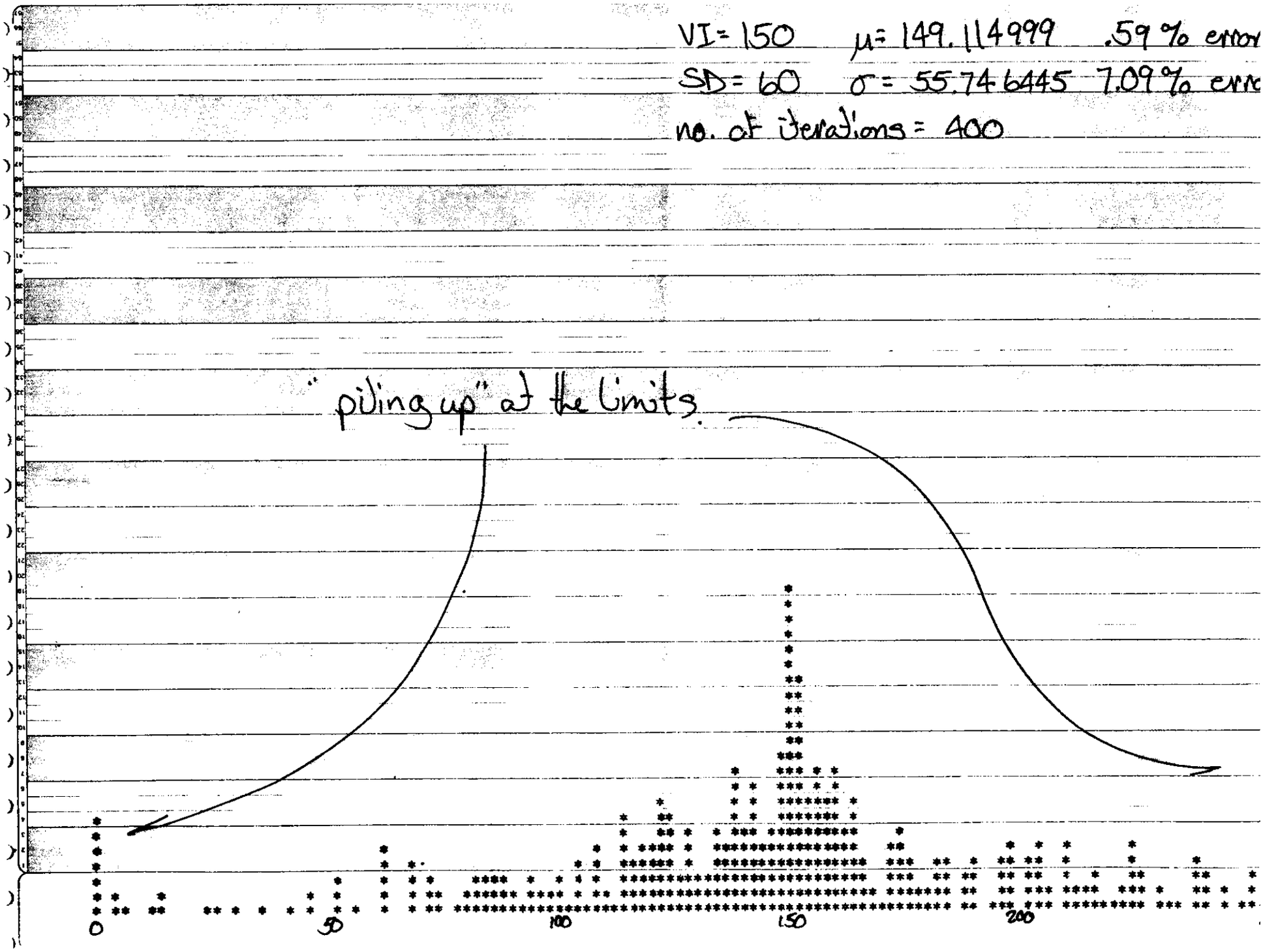

(continued From top of previous page)


VI = 127 μ = 127.279999 .22
SD = 63 σ = 55.925786 11.22
no. of iterations = 1000
each star indicates the occurrence of a event, or one sample.
"piling up" at the limits.
0
50
100
150
200
25

VI = 150 μ = 149.114999 .59 % error

SD = 60 σ = 55.746445 7.09 % error

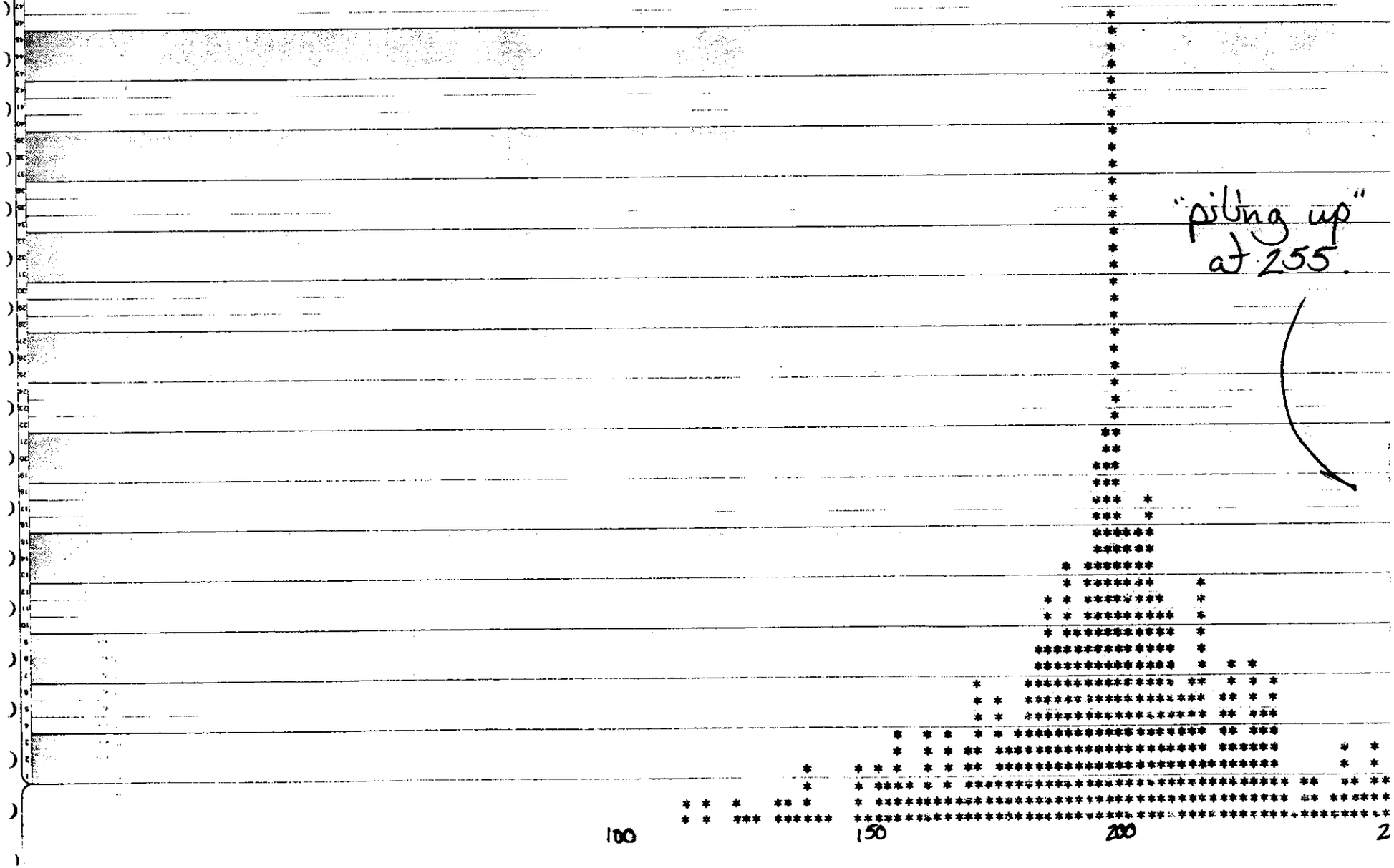
no. of iterations = 400



VI = 200 μ = 200.075999 .038% error

SD = 30 σ = 27.241564 9.195% error

no. of iterations = 500



Comments

on

The VI Schedule of Reinforcement Program

prepared for

Dr. Daniel W. Scott III

CSCI 488

by

Daniel Paul Long

Oct. 6, 1979

Comments

on

The VI Schedule of Reinforcement Program

During the development phase of an algorithm to generate randomly distributed normal values, I was going to use either the Polar method or the Direct method to generate the needed values in real time. To implement either method would require the ability to perform transcendental functions on the computer system. Initially rejecting the possibility of writing my own floating-point package to do these, I turned to Dr. Adams to use--pirate--parts of his M6800 BASIC interpreter. He told me that it would be too difficult to pull the necessary floating-point functions out of his interpreter and that I might try using the MP-N Calculator Interface, made by SWTPC, as an on-line numerical processor. I bought the MP-N with a school purchase order, assembled it, and implemented it using SWTPC-supplied software. By observation, I could see that the needed operations (eg. sinX and $\sqrt{X}$) would take more time to execute the distribution algorithms than the real time situation would allow.

One alternative I pursued was to use the MP-N to generate the values and store them in tables after the session was defined by the user, but before the actual session began. Each unit--chamber--would have its part of the table conform to the user-specified mean time (VI) and dispersion for that unit. I would have reserved 1K of RAM to contain all the data and would dynamically suballocate the 1K of

memory depending on the number of units to be served. But, since I orginally was to use the MP-N partly for its assumed speed, I looked for other, more elegant, means of solving my problem.

By the time I decided not to use the MP-N, I already had coded and assembled an early version of VISORP using the interface. Although I did not pursue it to fruition, I gained a little experience with using a numerical processor as a peripheral device and with writing and using floating-point routines.

Out of a meeting with my faculty advisor, Dr. Scott, I developed the algorithm described in the paper Functional Description of the VI Schedule of Reinforcement Program and implemented it with the M6800 ALC subroutine DEVIAT, in VISORP. In brief, I used a scaled, general static data base, or table, containing potential (I say potential because only after they are randomly assigned signs do they realize the following characteristics) randomly distibuted normal values with a mean of zero and with three standard deviations out from the mean occurring at 127. This data base was generated by the PL/I program NORMHEX using the Direct method of generating the above described distribution of values; see the NORMHEX source listing for the method. The data base is static in that its contents never changes. By knowing the dispersion characteristics of the table, I can quickly generate, in

DEVIAT, the variable time intervals required by each unit for any integer combination of mean (VI) and dispersion (SD) with the upper and lower bounds of 0,255 and 0,85, respectively. The last I heard, Dr. Emmett Ogelsby, a neurophysiologist with the Texas College of Osteopathic Medicine, was using my--this--solution to the same problem I was having, by using the DEVIAT subroutine and the associated data base in a program he was working on. With his satisfaction and my testing (see the paper Testing of The VI Schedule of Reinforcement Program), I am convinced of its reliability, suitability, and accuracy in carrying out its function.

Another evolving aspect of my work concerned how I structured the working storage. I originally had the data that was associated with each unit (eg. mean, dispersion, and maximum number of reinforcements allowed) contained in one large two-dimensional matrix. One dimension was the unit number; the other dimension was the characteristic, or particular item, of that unit. I came upon instances where I could reduce memory requirements and/or critical execution time (at this stage, I had no idea how long the worst case execution time would be and was anxious that all of my efforts would be in vain) by taking a particular item out of the matrix and making it a vector. I found that I was slowly, item-by-item, taking the one matrix completely apart and creating several separate vectors. Later, when I took the data structures class,

I recognized the indexing equations used to find elements in two-dimensional arrays that I had developed on my own for use in accessing my matrix's elements--a learning déjà vu.

When I took the computer systems analysis class, we discussed how to approach the solution to an information systems problem. I found that my faculty advisor had guided me along very much the same lines as our book and the instructor were delineating, such as the early and ongoing interaction between the analyst and user in a verbal and written description of the problem and its solution.

I had to take into consideration the limits of hardware when the eventual user, Dr. Ernie Harrell, decided that instead of a CRT terminal, like the ADM 3-A, he wanted to use a hardcopy terminal--the Decwriter LA36--in the computer system. The rub was that the maximum baud rate on the Decwriter was 300 which meant that it would take 33 milliseconds for each character to be printed as opposed to the much faster rates available on a CRT terminal. During a session, an event would occur (ie. the rat eating all of its allotted food pellets) where the program needed to notify the human operator to perform the manual task of removing the rat from the operant chamber. I originally wrote VISORP to print the corresponding unit number where the event occured, ring the terminal's bell, and skip to the next line on the CRT when this happens. Since it

takes 33 milliseconds to print just one character on the Decwriter and roughly 14 milliseconds for worst case execution of a polling cycle in the dispatch part of VISORP with the probable doubling of service capacity to 16 units--a total of 47 milliseconds, and since the application inherently requires a maximum of 50 milliseconds for a worst case execution time, I felt it was too close because further modifications and additions to the program might be hindered by this low ceiling. To solve the problem, I built a circuit board that would interface with the computer and alert the operator visually (LEDs) and aurally (piezo-electric transducer), that required only an inconsequential amount of time to use (see the paper: The Presettable Interrupt Frequency Generator/ Audio-Visual Indicator Board).

By keeping the interface between the computer and the operant behavior chambers simple, and delegating normally hardware functions to the software, the expense of hardware is reduced and the generality, hence versatility, of hardware is increased. The PIO board (see the paper: The Parallel Input/ Output Board) does no processing or storage of information-- the inputs are not latched or debounced and the outputs are not driven with one-shots. By using the interrupt frequency of 20 Hz as the base for timing hardware needs, the inputs are debounced and are read often enough not to require latching; the outputs are kept on just long enough to trigger the feeder mechanisms' coils

and there is no need to protect the would-be one-shots from glitches in the power supply with the use of filter capacitors.

For a long time, I worried that if more than one feeder coil were on at a time, the extra load on the power supply would blow the supply's fuse and spoil the session in progress. The supply's power rating and the load of each coil were such that the power supply could only handle the load of one coil at a time, even though the load lasts only 50 milliseconds per triggering. I asked Dr. Harrell if his present discrete apparatus (that the VISOR system would replace) experienced this problem and he told me that it did not. Either the probability of more than one rat receiving a reinforcement at the same time is extremely low or the short load duration is so short that the power supply can handle more than its stated load capacity, provided the higher loads are only for a short period of time. I quenched my worry by writing a short (17-byte) code segment that acted like a buffer, allowing only one of the coil output lines to be on at a time.

When I started assembling the first version of VISORP back in the fall of 1978, Dr. Adams's M6800 assembler was not available, so I used the cross-assembler on the IBM 360. To backup the VISORP card deck in the event that the cards were dropped and shuffled, damaged, or lost, I created a disc file containing the VISORP ALC. With

JCL, every subsequent VISORP assembly job deleted the old disc file, replaced it with the new instream data set, and then executed the cross-assembler using the new disc file as input. This way, I always maintained a backup copy of the current version of VISORP on disc. I wasted a couple of weeks, early on, by trying to get the IBM utility IEBUPDTE to work. I wanted to make modifications directly on the disc file instead of changing the card deck and reading all of it into the system every time I required a new assembly. I could make deletions and replacements of records, but not insertions--they came out as appendages.

Using the cross-assembler meant that I had to reenter the object module--over 1K--at the ADM terminal, by hand, every time I made a major or several minor modifications to VISORP. To make this task easier, I wrote and used an ALC program that would allow me to print and enter hex data at the terminal in the same format as the object file listing option of the cross-assembler. This program (the listing can be found accompanying this text; it is called FORMATTED CHANGE/DUMP) made it easier to check discrepancies between the actual contents of memory and the supposed object file listing of the cross-assembler.

In May, during the interim between spring and summer sessions, I finished debugging VISORP. I owe the short amount of time it took to actually get the system working once it was finally put together, to my frequent manual

simulations of the various subprograms in VISORP. When I did not have any pressing changes to make to the program, I would use the backs of old printouts to write out the results each instruction had on the effected variables through a theoretical run of sections of code, even though the section was not particularly suspect. I used this practice throughout software development and preventative, aggresive, and adventurous debugging.

Despite my testing, I still worry that it may not work either partially or wholly, because it has not been proven in actual use. I feel that I did my best to test it and that, at least within my scope of testing, it works. But, Dr. Harrell has never had to use my session parameter definition syntax, only one chamber has been used to test it, and no rats have even gotten near it. I feel like a composer who has written a score that he's really proud of, yet no one has, or may ever, hear it. Am I secure enough to let anyone hear it? I can tell you all the great things it does, but until it is used, this might as well be a hoax.

Recently, I have been interviewing for jobs, and twice the interviewer has asked what my best--favorite, most valuable--class was. I told them about the special problems courses I have been taking persuant to VISOR.

FORMATTED CHANGE/DUMP

program listing

prepared for

Dr. Daniel W. Scott III
CSCI 488

by

Daniel Paul Long
January 12, 1979

```
00001 ********************************************************************************
00002 *                                                                              *
00003 *                                                                              *
00004 *                          FORMATTED CHANGE/DUMP                               *
00005 *                                                                              *
00006 *       1/12/79            DANIEL PAUL LONG              N.T.S.U.             *
00007 *                          M6800 ASSEMBLY LANGUAGE                             *
00008 *                          158 BYTES                                           *
00009 *                                                                              *
00010 *                                                                              *
00011 *   THIS PROGRAM IS INTENDED TO BE USED IN AN M6800 MICROCOMPUTER SYSTEM        *
00012 * WITH A MIKBUG MONITOR. BY USING MIKBUG SUBROUTINES AS MUCH AS POSSIBLE,      *
00013 * THIS PROGRAM PROVIDES THE USER WITH 2 MEMORY FUNCTIONS: 'C' CHANGE &         *
00014 * 'D' DUMP.                                                                    *
00015 *   INPUT AND OUTPUT OF THIS PROGRAM HAVE THE SAME FORMAT AS THE 'LISTING      *
00016 * OF OBJECT FILE' DUMP OPTION OF THE M6800 ASSEMBLER ON THE IBM 360. THE       *
00017 * PURPOSE OF THIS PROGRAM IS TO MAKE DEBUGGING EASIER AND TO SAVE TIME IN      *
00018 * ENTERING A PROGRAM.                                                          *
00019 *                                                                              *
00020 *  *                                                                           *
00021 *   AT THE BEGINNING OF THE PROGRAM, A NUMBER SYMBOL (#) WILL APPEAR AT THE    *
00022 * BEGINNING OF THE NEXT LINE, WHEREUPON A 'C' OR 'D' SHOULD BE ENTERED.       *
00023 *                                                                              *
00024 *                                                                              *
00025 * I) IF 'C' IS TYPED:                                                          *
00026 *   A) THE CURSOR MOVES TO THE BEGINNING OF THE NEXT LINE.                     *
00027 *   B) ENTER THE 4-CHARACTER HEX ADDRESS AT WHICH YOU WISH TO START ENTERING   *
00028 * DATA. THE TERMINAL INSERTS 2 SPACES AFTER THE ADDRESS.                       *
00029 *   C) ENTER THE NEW 2-CHARACTER HEX DATA TO BE STORED AT THIS LOCATION. THE   *
00030 * TERMINAL INSERTS A SPACE AFTER THE DATA.                                     *
00031 *   D) CONTINUE ENTERING DATA AS IN STEP C. ALL SPACING IS DONE BY THE         *
00032 * TERMINAL. AFTER STEP C HAS BEEN REPEATED 16 TIMES, THE TERMINAL PRINTS AN    *
00033 * ASTERISK, BEEPS, SKIPS TO THE BEGINNING OF THE NEXT LINE, PRINTS THE        *
00034 * ADDRESS OF THE NEXT LOCATION (PREVIOUSLY APPEARING ADDRESS + $10), AND      *
00035 * SKIPS 2 SPACES.                                                              *
00036 *   E) STEP D IS REPEATED INDEFINITELY UNTIL ONE OF THE FOLLOWING OCCURS:      *
00037 *     1) '*' IS ENTERED IN THE FIRST CHARACTER POSITION OF DATA, WHEREUPON     *
00038 *        THE PROGRAM CONTINUES FROM STEP A.                                    *
00039 *     2) '.' IS ENTERED IN THE FIRST CHARACTER POSITION OF DATA, WHEREUPON     *
00040 *        THE PROGRAM CONTINUES FROM ITS BEGINNING.                             *
00041 *     3) ANY NON-HEX CHARACTER IS ENTERED AS DATA, (OTHER THAN THE 2 ABOVE     *
00042 *        EXCEPTIONS), WHEREUPON CONTROL IS TRANSFERED TO MIKBUG.               *
00043 *     4) DATA IS ATTEMPTING TO BE STORED INTO NONEXISTENT MEMORY OR ROM,      *
00044 *        WHEREUPON A '?' IS PRINTED AND CONTROL IS TRANSFERED TO MIKBUG.       *
00045 *                                                                      CONT.   *
00046 ********************************************************************************
```

*Begin execution at $F60$_{16}$

```
00048 ********************************************************************************
00049 *                                                                              *
00050 *  II) IF 'D' IS TYPED:                                                        *
00051 *    A) ENTER THE BEGINNING 4-CHARACTER HEX ADDRESS.                           *
00052 *    B) THE TERMINAL INSERTS A SPACE AFTER THIS ADDRESS.                       *
00053 *    C) ENTER THE ENDING 4-CHARACTER HEX ADDRESS.                              *
00054 *    D) THE CURSOR MOVES TO THE BEGINNING OF THE NEXT LINE.                    *
00055 *    E) THE TERMINAL PRINTS THE BEGINNING ADDRESS AND 2 SPACES.                *
00056 *    F) DATA FROM THE NEXT 16 LOCATIONS ARE PRINTED ON THE SAME LINE, EACH     *
00057 *  FOLLOWED BY A SPACE.                                                        *
00058 *    G) AN ASTERISK IS PRINTED, THE TERMINAL BEEPS, STEP D IS REPEATED, THE    *
00059 *  ADDRESS OF THE NEXT LOCATION (PREVIOUSLY APPEARING ADDRESS + $10) IS        *
00060 *  PRINTED, THE TERMINAL INSERTS 2 SPACES, AND CONTINUES FROM STEP F UNTIL ALL *
00061 *  DATA HAS BEEN PRINTED UP TO THE ENDING ADDRESS.                             *
00062 *    H) CONTINUE FROM THE BEGINNING OF THE PROGRAM.                            *
00063 *                                                                              *
00064 *  III) IF NEITHER IS TYPED:                                                   *
00065 *    THE TERMINAL PRINTS A '?' AND CONTINUES FROM THE BEGINNING OF THE PROGRAM.*
00066 *                                                                              *
00067 *                                                                              *
00068 * (NOTE: IF NON-HEX CHARACTERS ARE ENTERED DURING ANY OF THE ADDRESS INPUTTING *
00069 *  PHASES, A '?' IS PRINTED AND CONTROL IS TRANSFERED TO MIKBUG.)               *
00070 *                                                                              *
00071 *                                                                              *
00072 ********************************************************************************


00074                          NAM   FORMATTED CHANGE/DUMP
00075                          OPT   MEMORY


00077                  * ALL VARIABLES ARE IN MIKBUG'S RAM.

00079 A004       ENDA  EQU   $A004
00080 A00C       XHI   EQU   $A00C
00081 A00F       TW    EQU   $A00F


00083                  * SR'S & AN ENTRY POINT INTO MIKBUG (LOAD19).

00085 E040       LOAD19 EQU  $E040
00086 E047       BADDR  EQU  $E047
00087 E055       BYTE   EQU  $E055
00088 E07E       PDATA1 EQU  $E07E
00089 E0AA       INHEX  EQU  $E0AA
00090 E0BF       OUT2H  EQU  $E0BF
00091 E0C8       OUT4HS EQU  $E0C8
00092 E0CC       OUTS   EQU  $E0CC
00093 E1AC       INEEE  EQU  $E1AC
00094 E1D1       OUTEEE EQU  $E1D1
```

```
00096                    ********************************************************************************
00097                    *    CODE SEGMENT WHERE OPTION DESIRED IS ENTERED.  'C' CHANGE  'D' DUMP       *
00098                    ********************************************************************************

00100  0F60                      ORG    $0F60
00101  0F60  8D  59      SELECT  BSR    NXTLIN          PRINT CR, LF
00102  0F62  86  23              LDA A  #'#
00103  0F64  BD  E1D1            JSR    OUTEEE          PRINT NUMBER SYMBOL
00104  0F67  BD  E1AC            JSR    INEEE           INPUT FUNCTION

00106  0F6A  81  43              CMP A  #'C
00107  0F6C  27  08              BEQ    ENTOBJ          CHANGE MEMORY
00108  0F6E  81  44              CMP A  #'D
00109  0F70  27  50              BEQ    PRTOBJ          DUMP MEMORY
00110  0F72  86  3F              LDA A  #'?             NEITHER
00111  0F74  BD  E1D1            JSR    OUTEEE
00112  0F77  20  E7              BRA    SELECT
```

```
00114                    ********************************************************************************
00115                    *    DATA IS STORED INTO MEMORY IN THIS SEGMENT                                *
00116                    ********************************************************************************

00118  0F79  8D  40  ENTOBJ BSR  NXTLIN        PRINT CR, LF
00119  0F7B  BD  E047       JSR  BADDR         BUILD ADDRESS
00120  0F7E  BD  E0CC       JSR  OUTS          PRINT SPACE

00122  0F81  C6  10  INLINE LDA B #16          16 BYTES/LINE

00124  0F83  BD  E0CC  INBYTE JSR  OUTS        PRINT SPACE
00125  0F86  BD  E1AC       JSR  INEEE         INPUT FIRST CHAR.
00126  0F89  81  2A         CMP A #'*
00127  0F8B  27  EC         BEQ  ENTOBJ        IS AN ASTERISK

00129  0F8D  81  2E         CMP A #'.
00130  0F8F  27  CF         BEQ  SELECT        IS A PERIOD
00131  0F91  BD  E0AC       JSR  INHEX+2       CHECK IF HEX
00132  0F94  37             PSH B              SAVE B REGISTER
00133  0F95  BD  E057       JSR  BYTE+2        INPUT SECOND CHAR
00134  0F98  33             PUL B              RESTORE B REGISTER
00135  0F99  A7  00         STA A 0,X          CHANGE MEMORY
00136  0F9B  A1  00         CMP A 0,X
00137  0F9D  27  03         BEQ  *+2+3         DID CHANGE
00138  0F9F  7E  E040       JMP  LOAD19        NOT CHANGED
00139  0FA2  08             INX
00140  0FA3  5A             DEC B
00141  0FA4  26  DD         BNE  INBYTE        SAME LINE

00143  0FA6  FF  A00C       STX  XHI
00144  0FA9  8D  0B         BSR  ASTER         PRINT ' *',CR, LF
00145  0FAB  CE  A00C       LDX  #XHI
00146  0FAE  BD  E0C8       JSR  OUT4HS        PRINT ADDRESS
00147  0FB1  FE  A00C       LDX  XHI
00148  0FB4  20  CB         BRA  INLINE        NEXT LINE OF INPUT
```

```
00150                    ********************************************************************************
00151                    *    OUTPUT SR'S - ASTER PRINTS A STAR AND NXTLIN PRINTS CR & LF                *
00152                    ********************************************************************************

00154   0FB6   CE 0FE9   ASTER  LDX   #SPCSTR        PRINT ' *', BELL, CR LF
00155   0FB9   20 03             BRA   *+2+3
00156   0FBB   CE 0FFC   NXTLIN LDX   #CRLF          PRINT CR, LF
00157   0FBE   BD E07E           JSR   PDATA1

00159   0FC1   39                RTS
```

```
00161                    ********************************************************************************
00162                    *    DATA IS PRINTED FROM MEMORY IN THIS SEGMENT                               *
00163                    ********************************************************************************

00165  0FC2  BD  E047  PRTOBJ JSR   BADDR      BUILD BEGINNING ADDRESS
00166  0FC5  FF  A00F         STX   TW
00167  0FC8  BD  E0CC         JSR   OUTS       PRINT SPACE BETWEEN ADDRESSES
00168  0FCB  BD  E047         JSR   BADDR      BUILD ENDING ADDRESS
00169  0FCE  08               INX
00170  0FCF  FF  A004         STX   ENDA
00171  0FD2  8D  E7           BSR   NXTLIN     PRINT CR, LF

00173  0FD4  CE  A00F  OUTLIN LDX   #TW
00174  0FD7  BD  E0C8         JSR   OUT4HS     PRINT ADDRESS
00175  0FDA  C6  10           LDA B #16        16 BYTES/LINE
00176  0FDC  FE  A00F         LDX   TW         RESTORE DATA ADDRESS

00178  0FDF  BD  E0CC  OUTBYT JSR   OUTS       PRINT SPACE
00179  0FE2  BD  E0BF         JSR   OUT2H      PRINT DATA
00180  0FE5  BC  A004         CPX   ENDA
00181  0FE8  27  0A           BEQ   ALLOUT     ALL DATA PRINTED

00183  0FEA  5A               DEC B
00184  0FEB  26  F2           BNE   OUTBYT     SAME LINE

00186  0FED  FF  A00F         STX   TW         SAVE DATA ADDRESS
00187  0FF0  8D  C4           BSR   ASTER      PRINT ' **, CR, LF
00188  0FF2  20  E0           BRA   OUTLIN     NEXT LINE OF OUTPUT

00190  0FF4  8D  C0  ALLOUT BSR   ASTER       PRINT ' **, CR, LF
00191  0FF6  7E  0F62         JMP   SELECT+2
```

```
00193   0FF9   20          SPCSTR FCC   2, *          CHAR: ' *'
        0FFA   2A
00194   0FFB   07                 FCB   $7            BELL
00195   0FFC   0D          CRLF   FCB   $D,$A,4       CARRIAGE RETURN, LINE FEED CHARACTERS
        0FFD   0A
        0FFE   04

00197                      END

TOTAL ERRORS    0
```

LISTING OF OBJECT FILE

```
OF60  8D 59 86 23 BD E1 D1 BD E1 AC 81 43 27 0B 81 44 *
OF70  27 50 86 3F BD E1 D1 20 E7 8D 40 BD E0 47 BD E0 *
OF80  CC C6 10 8D E0 CC BD E1 AC 81 2A 27 EC 81 2E 27 *
OF90  CF 8D E0 AC 37 BD E0 57 33 A7 00 A1 00 27 03 7E *
OFA0  E0 40 08 5A 26 DD FF A0 0C 8D 08 CE A0 0C 8D E0 *
OFB0  C8 FE A0 0C 20 CB CE 0F F9 20 03 CE 0F FC BD E0 *
OFC0  7E 39 8D E0 47 FF A0 0F 8D E0 CC BD E0 47 08 FF *
OFD0  A0 04 8D E7 CE A0 0F BD E0 C8 C6 10 FE A0 0F 8D *
OFE0  E0 CC BD E0 BF BC A0 04 27 0A 5A 26 F2 FF A0 0F *
OFF0  8D C4 20 E0 8D C0 7E 0F 62 20 2A 07 0D 0A 04 *
```

Documentation

on

The VI Schedule of Reinforcement Hardware

prepared for

Dr. Daniel W. Scott III

CSCI 490

by

Daniel Paul Long

The Parallel Input/Output Board

prepared for

Dr. Daniel W. Scott III

CSCI 490

by

Daniel Paul Long

December 9, 1979

The Parallel Input/Output Board

The Parallel Input/Output (PIO) board is a parallel input/output interface between the outside world and a Southwest Technical Products Corporation (SWTPC) Motorola M6800-based computer system. The input and output functions, each having 8 two-state channels are combined on one printed circuit board to be inserted onto an input/output (I/O) port in a SWTPC computer system.

Functional Description

Inputs

Connection between the board's inputs and external, user-supplied switches must be provided in the form of a nine-conductor (8 inputs and ground) cable with a male 10-pin Molex connector on the computer system's end (see the parts layout diagram and the external wiring diagram following this text).

Each input channel is defined by either being shorted to ground (closed switch) or floating (open switch). When shorted to ground, the bit position corresponding to that channel is read as a zero; when floating, the bit is read as a one. In reference to the external wiring diagram, if switches 8, 7, 6, 4, 3, and 1 were open and switches 2 and 5 were closed, the ALC instruction

```
LDA A DATAA   READ DATA REGISTER A
```

would result in the A accumulator of the M6800 having the value 11101101_2.

To obtain the described input (and output) function, the Peripheral Interface Adapter (PIA)--a Motorola M6820 integrated circuit--must be initialized. To initialize the PIA, execute the following instructions.

```
LDA A #$FF
LDA B #4
CLR   DATAA   DEFINE INPUT
STA B CONTRA
STA A DATAB   DEFINE OUTPUT
STA B CONTRB
STA A DATAB   RESET OUTPUTS
```

Outputs

The outputs of the PIO board require an external, user-supplied DC power supply rated at no more than 60 volts at a maximum of 2 amperes for a resistive load. Connection between the board's outputs and external, user-supplied driven devices must be provided in the form of a ten-conductor (8 outputs, V_{EE}, and G_{EE}) cable with a male 10-pin Molex connector on the computer system's end (see the parts layout diagram and external wiring diagram).

Each output channel can handle up to 50 watts of continuous power disspation and can either be "on" or "off." The "on" state will provide the external power supply's voltage, V_{EE}, at the output channel, while the "off" state will float the output channel.

To produce an "on" state at a particular output channel, store a zero at the bit position corresponding to that channel; for an "off" state, store a one at that bit position. In reference to the external wiring diagram, to turn devices 7, 4, 3, 2, and 1 off and turn devices 8, 6, and 5 on, execute the following ALC instructions.

```
LDA A #%01001111
STA A DATAB   STORE INTO DATA REGISTER B
```

Theory of Implementation

Inputs

A simple parallel interface between the computer system's 8-bit data buss and an 8-channel socket transfers data, in parallel, from the socket to the buss.

Outputs

Data is transfered from the computer system's data buss to an 8-channel socket through various interface stages including one stage of electrical isolation between the computer system's power supply and the output devices' power supply.

Practicalities of Implementation

Common Functional Aspects

Both the input and output circuits of the PIO are

interfaced to the SWTPC 6800 buss by the same PIA. The inputs use the A side of the PIA; the outputs use the B side.

The inputs use the +5VDC power supply from the on board voltage regulator. The computer's side of the output circuitry (from the light-emitting-diodes of the optical isolators, back) uses the +5VDC from the same voltage regulator as the inputs do.

Inputs

Each input channel is held high by a pull-up resistor connected to +5VDC. A floating input line (open switch) lets the pull-up resistor define the input as a one; a grounded input (closed switch) pulls the line low, defining the input as a zero. These defined states are then interfaced to the system's data buss through the A side of the PIA.

Outputs

Coming out of the PIA, each output channel is inverted through a TTL inverter. It is then connected through a current limiting resistor to the anode of an LED in an optical isolator; the LEDs cathode is tied to ground. The emitter of the isolator's output phototransistor is tied to the external DC power supply's ground, G_{EE}. The phototransistor's collector is connected through

a current limiting load resistor to the base of a power transistor. The power transistor's emitter is tied to the positive side of the external DC power supply, V_{EE}; its collector is connected to the driven device (see the external wiring diagram).

In summary, a one at the PIA's input--hence, output--turns the LED off so that the phototransistor is not biased, which means that the power transistor is not biased either, and no current can flow through the driven device; a zero turns the LED on so that the phototransistor is biased, which, in turn, biases the power transistor, and allows current to flow through the driven device.

Testing Rationale

Inputs

With the PIO board in place on an I/O port of the computer system's motherboard, the system monitor is used to initialize and display the contents of the A data register of the PIA. By selectively bringing the input channels to ground by inserting one end of a jumper wire into the ground slot, the other end into the input slots of the female Molex connector, the A data register of the PIA should reflect the manipulation of the jumper wire.

ones in it. When an input channel was jumpered, the bit position had a zero in it. In all tests of individual channels, and then in tests of various combinations, the A data register reflected what was being done with the jumper(s).

Outputs

As planned, the coil would click on and the oscilloscope would indicate V_{EE} across the coil when a zero was stored in the corresponding bit position of the B data register; click off and the oscilloscope would indicate no voltage across the coil when a one was stored. I tested each output channel in the same way with similar results.

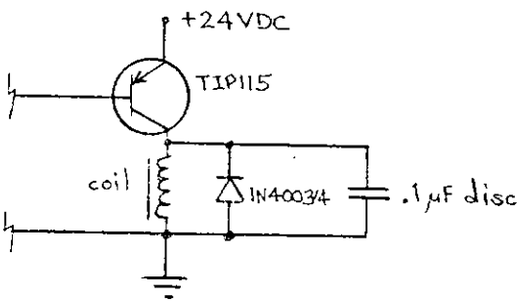
Comments

Outputs

With the coil disconnected and just the inductive load of the oscilloscope, I found a faulty power transistor. When the output (collector) of this transistor was supposed to be low, the potential difference between G_{EE} and the collector would slowly increase up to 24V of the external power supply. This condition was unique to this output channel. With the coil connected, it would always stay on. Replacing the transistor fixed the problem.

As I continued to test the circuit, some of the output channels could not be brought low. By swapping out power transistors and swapping the optical isolator packages (there are 2 of them in the circuit), I pinpointed the problem to two more shorted power transistors and at least one burned out channel in each of the isolator packages. I later learned that the cause of these burn-outs was that I was not protecting the outputs from the driven device reverse biasing the transistors. When the feeding mechanism's coil is energized, a piston inside the coil is electromagnetically forced up, causing a food pellet to fall out of a reservoir. When the coil is turned off, the piston falls back down by force of gravity. As it falls down, the magnetic field of the coil collapses and produces current through the coil windings in the opposite direction of the current that energizes the coil to force the piston up. These two effects reverse bias the power transistor and phototransistor on the PIO board for a short period of time. My repetitive testing procedure eventually destroyed the aforementioned components. I replaced the bad transistors and isolators and connected across the coil, a diode and capacitor to prevent the coil from damaging the transistors. The subsequent devices that are operated by this board should not reverse bias the outputs of the PIO board. Since the PIO does not protect its own outputs, the

driven device must do so. Here is part of the output section of the PIO with a coil as the driven device and with reverse biasing protection:

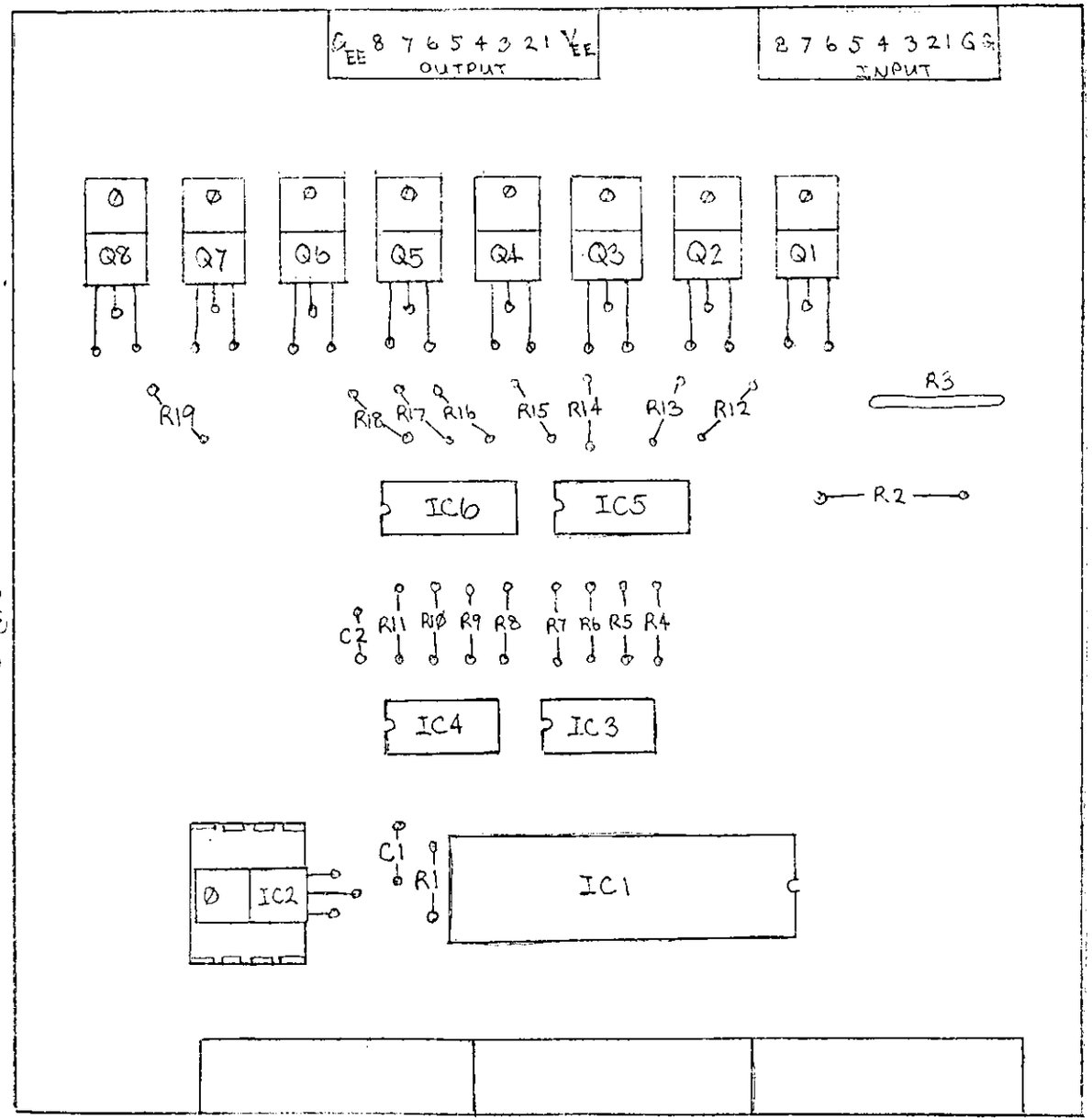


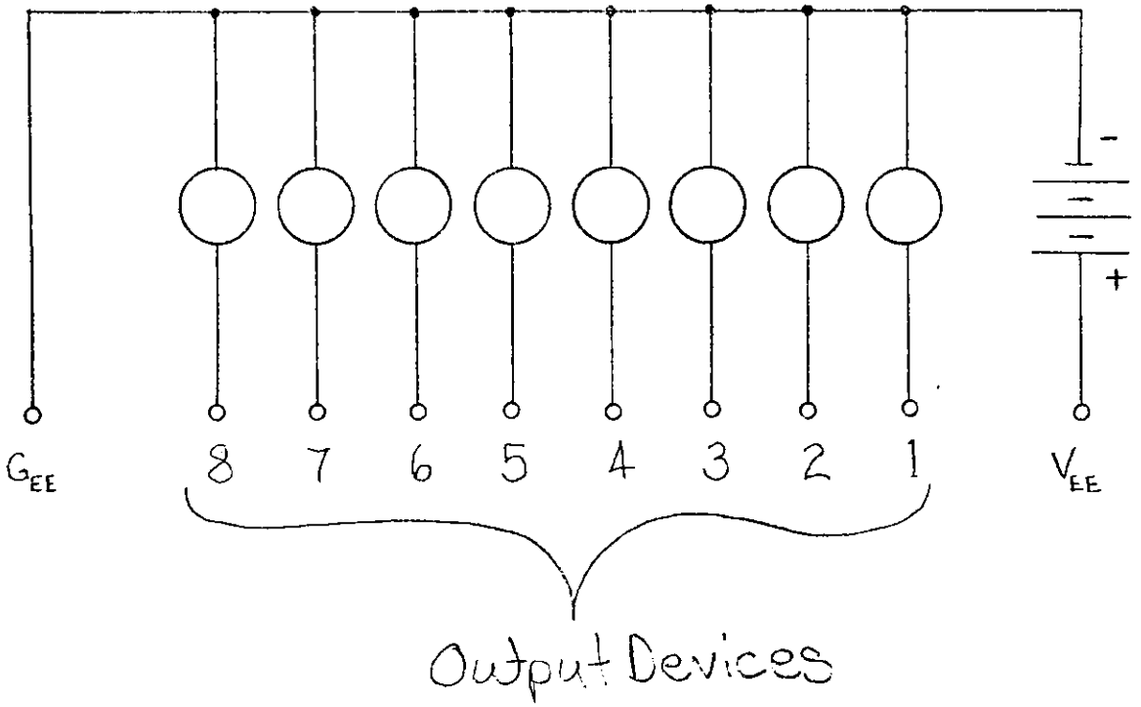
Layout and Parts List

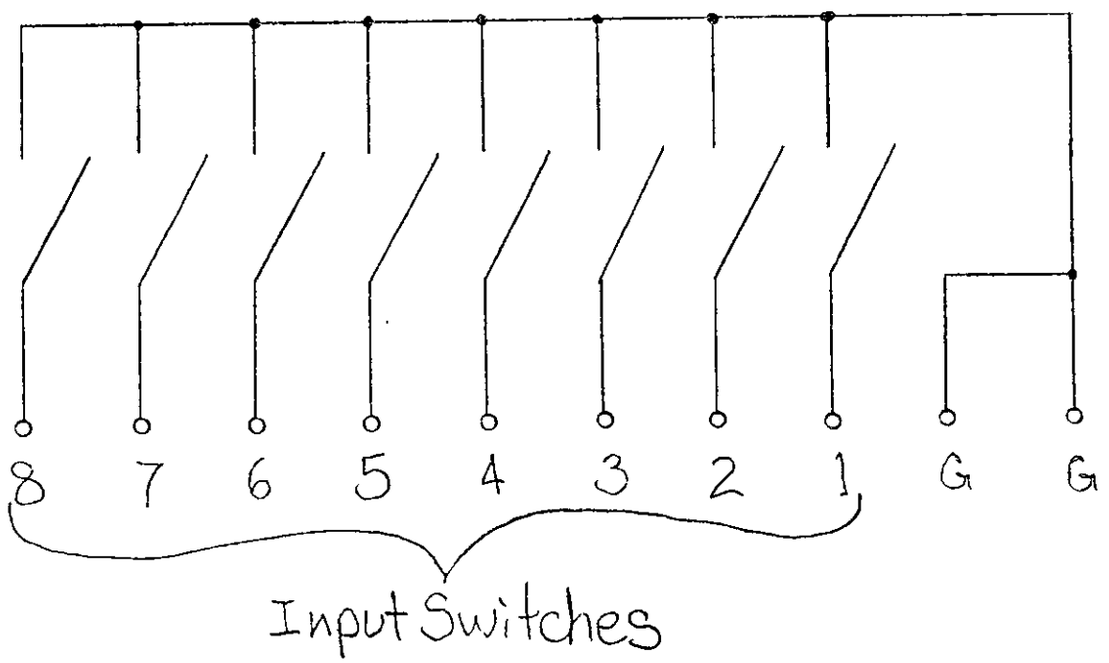
Parts List

R1,R2 1K ¼ watt resistors
R3 1K 7-resistor pack
R4-R11 300 Ω ¼ watt resistors
R12-R19 6K ¼ watt resistors
C1 1 μF electrolytic cap.
C2 .1μF capacitor
Q1-Q8 TIP115 power transistors
IC1 6820 PIA
IC2 7805 voltage regulator
IC3,IC4 7404 hex inverters
IC5,IC6 ILQ-74 opto-isolators

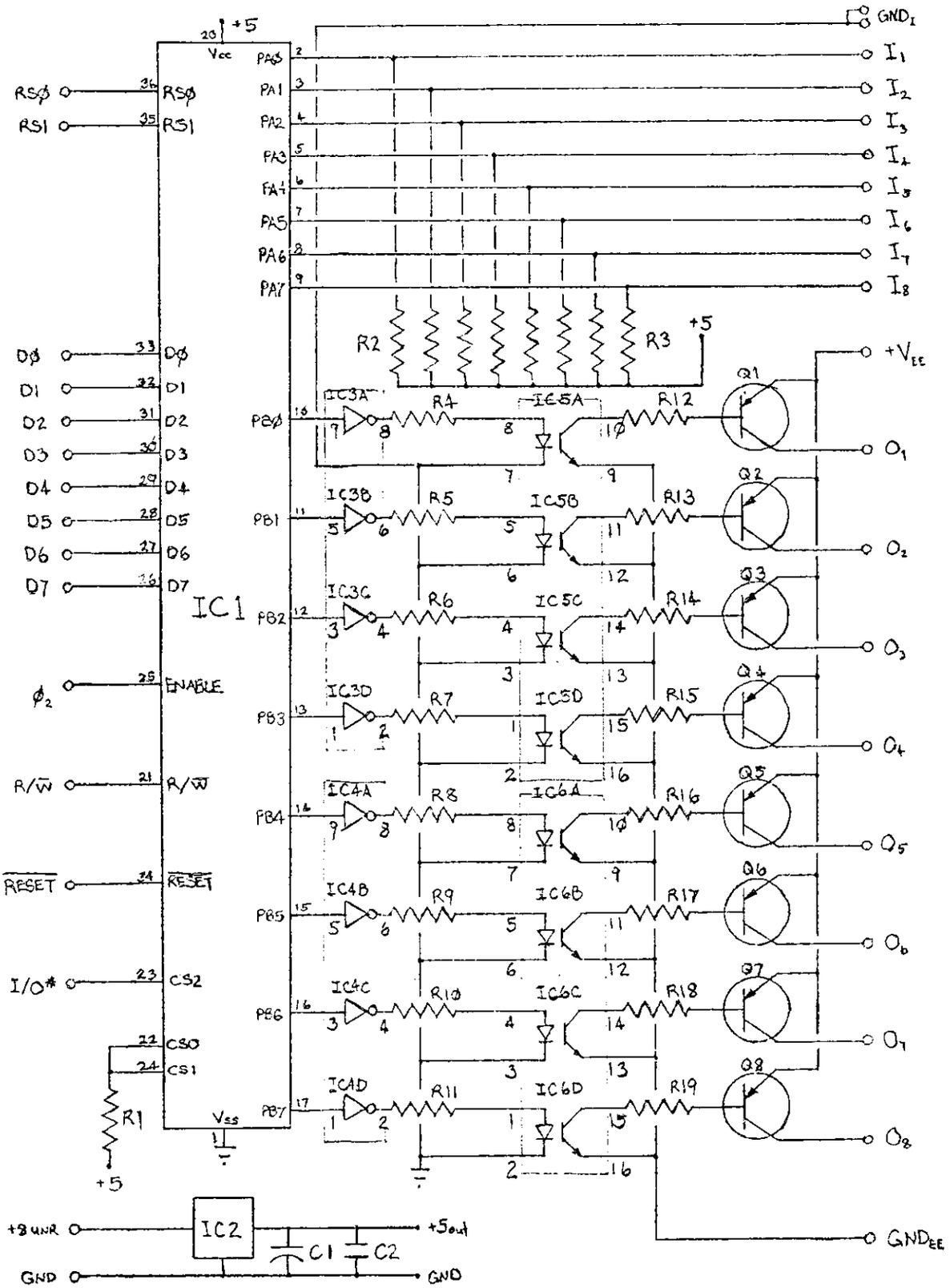
Misc: 5 female Molex 10-pin sockets, 2 14-pin sockets, 2 16-pin sockets, 1 40-pin socket, TO-5 heatsink, pc board, hardware.







External Wiring Diagram


+5
Vcc
RSØ
RS1
DØ
D1
D2
D3
D4
D5
D6
D7
IC1
ENABLE
R/W
RESET
I/O*
CS2
CSØ
CS1
Vss
R1
PAØ
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PBØ
PB1
PB2
PB3
PB4
PB5
PB6
PB7
R2
R3
IC3A
IC3B
IC3C
IC3D
IC4A
IC4B
IC4C
IC4D
IC5A
IC5B
IC5C
IC5D
IC6A
IC6B
IC6C
IC6D
R4
R5
R6
R7
R8
R9
RIØ
R11
R12
R13
R14
R15
R16
R17
R18
R19
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
GNDI
I1
I2
I3
I4
I5
I6
I7
I8
+VEE
O1
O2
O3
O4
O5
O6
O7
O8
GNDEE
+8 UNR
IC2
C1
C2
+5out
GND
GND

The Presettable Interrupt Frequency Generator/ Audio-Visual Indicator Board

prepared for

Dr. Daniel W. Scott III
CSCI 490

by

Daniel Paul Long
November 16, 1979

The Presettable Interrupt Frequency Generator/ Audio-Visual Indicator Board

The Presettable Interrupt Frequency Generator (PIFG) provides a wide range of interrupt frequencies (7 Hz to 19 KHz) applied to the $\overline{IRQ}$ (Interrupt ReQuest) control line on the computer system buss. The Audio-Visual Indicator (AVI) provides visual indication in the form of a light-emitting-diode (LED) connected to each of the eight bit positions of the peripheral chip's B data register. Audio indication comes from a piezo-electric oscillator that beeps whenever the above data register is written into. The PIFG/AVI board combines these two unrelated functions on one printed circuit board to be inserted onto an input/output (I/O) port in a Southwest Technical Products Corporation (SWTPC) Motorola M6800-based computer system.

Functional Description

PIFG

The PIFG obtains its starting frequency from any of the five baud lines on the buss: 110, 150, 300, 600, and 1200. The system buss actually provides 16 times the indicated baud rate (eg. the 110-baud line actually provides 1760 Hz). The board was assembled with a wire jumper connected to the 110-baud line, but this jumper

may be reconnected to any of the other baud lines as a means of coarse frequency adjustment. To divide the jumper selected signal down to the interrupt frequency required for the user's application, manually set the first eight rocker switches on the DIP switch package (see the parts layout diagram following this text). The number the user loads (sets) at the switch package is 255 minus the divisor. "For a divide-by-132, first find 255 minus 132=123. Convert 123 to binary=64 + 32 + 16 + 8 + 2 + 1."[1] This number in binary would be 01111011. The ninth switch is not used, and the tenth switch inhibits the interrupt requests altogether. By monitoring the $\overline{\text{IRQ}}$ control line with an oscilloscope while servicing the interrupts by software, the actual interrupt frequency may be calculated by observing the waveform on the scope.

To obtain the described function of the PIFG (and AVI), the Peripheral Interface Adapter (PIA)--a Motorola M6820 integrated circuit--must be initialized. To initialize the PIA, execute the following instructions.

```
LDA A #$FF
LDA B #$AD
CLR   CONTRB
STA A DATAB
STA B CONTRB
```

[1]Don Lancaster, TTL Cookbook(Indianapolis, Indiana: Howard W. Sams & Co., Inc., 1974), p. 239.

Modified operation of the board can be obtained by using alternate initialization procedures.

AVI

To turn on (and off) the AVI's LEDs and make the piezo-electric oscillator beep, write the data you want displayed into the B data register of the PIA. For instance, to turn on every other LED starting with LED1 (see the parts layout diagram), execute the following instructions.

```
LDA A #%01010101
STA A DATAB
```

This will result in the even-numbered LEDs being off and the odd-numbered LEDs being on.

The piezo-electric oscillator is activated whenever the B data register of the PIA is written into, so that the above store instruction would have caused it to beep. To change the duration of the oscillator's beep from a few milliseconds to about a second, adjust the micro-potentiometer located on the board (see the parts layout diagram). It is advisable to adjust this micro-potentiometer so that the duration is short and the sound is down to an inocuous chirp because the pitch and volume of the oscillator can become irritating.

Theory of Implementation

PIFG

A non-continuous, relatively wide range of periodic hardware interrupts is obtained by dividing a fixed frequency, already available on the system buss, by an integer divisor (this may be preset manually on the circuit board) via a hardware counting system. The resulting quotient is applied to the system's interrupt request control line.

AVI

A simple parallel interface between the computer system's 8-bit data buss and eight LEDs causes the LEDs to reflect the data that is addressed to them. By sensing when the interface is being addressed and written into, a sound is made to announce the fact that the data indicated by the LEDs has been updated.

Practicalities of Implementation

Common Functional Aspects

Both the PIFG and the AVI circuits are interfaced to the SWTPC 6800 buss by the same PIA. They also share the same +5 VDC power from the on-board voltage regulator.

PIFG

Two cascaded, synchronous 4-bit binary counters (74161s) divide the starting frequency (obtained, through a wire jumper, from any one of the five baud lines on the buss) by a divisor of from 1 to 255 (see the circuit schematic diagram following this text). The divisor is loaded into the counters by manually setting the first eight rocker switches on the DIP switch package to its 1's complement. The DIP switch package's tenth switch is used to inhibit the interrupts.

The output of the counters is a symmetrical square wave. This output is connected to CB1 on the PIA. When this signal at CB1 goes from high to low (transition synchronization is actually affected by the PIA's ENABLE input being connected to $\emptyset_2$ on the system buss), the $\overline{\text{IRQ}}$ line is brought low by the $\overline{\text{IRQB}}$ output of the PIA. The $\overline{\text{IRQ}}$ line remains low until either the reset button is pressed on the computer or a software read from the B data register of the PIA is executed--either case results in the $\overline{\text{IRQ}}$ line going high.

The $\overline{\text{IRQ}}$ line is connected to $\overline{\text{IRQB}}$ of the PIA with a jumper. Provisions are made to jump, instead, from $\overline{\text{IRQB}}$ to the $\overline{\text{NMI}}$ control line on the buss, providing non-maskable interrupts.

AVI

The LEDs of the AVI are on when there is a one in the corresponding bit position of the B data register of the PIA and, likewise, are off when there is a zero in the corresponding bit position. The anodes of the LEDs are all connected to the +5 VDC power supply (see the circuit schematic diagram). Each cathode is connected to the output of a TTL inverter (7404) through a resistor that limits the current passing through the LED. The input of each inverter is connected to a bit position of the B data register of the PIA.

The piezo-electric oscillator is activated whenever a software write into the B data register of the PIA is executed. The write causes CB2 on the PIA to go low then return high for one system clock cycle. CB2 is connected to the trigger of a one-shot (a type 555 integrated circuit in its monostable configuration) so that when CB2 goes low, the one-shot's output--connected to the piezo-electric oscillator--goes high and remains high for a variable amount of time from a few milliseconds to about a second. The duration is set by the micro-potentiometer on the board. The piezo-electric oscillator has internal solid-state circuitry so that the "one" from the one-shot's output causes it to oscillate at a constant 4.8 KHz for the duration of the pulse.

Testing Rationale

AVI

To test the AVI circuit, I wrote an assembly language program that initializes the PIA using the assembly language instructions in the PIFG Functional Description section, then waits for a character to be entered at the terminal that is interfaced with the computer system. The program writes the ASCII code value of that character into the B data register of the PIA and then loops back to wait for another character to be entered.

The effect that the micro-potentiometer has on the piezo-electric oscillator's beep duration can be heard by turning the micro-potentiometer as characters are entered at the terminal's keyboard. Every time a character is entered, the oscillator should beep.

PIFG

The test program also had code in it to test the PIFG circuit. Upon each interrupt request, control is transfered to a small interrupt servicing subroutine that resets the $\overline{IRQ}$ line to high by reading from the B data register of the PIA, then returns from the interrupt request routine, back to the main program that tests the AVI circuit.

While the program is running, the $\overline{IRQ}$ waveform can be observed on an oscilloscope whose vertical input is connected to the $\overline{IRQ}$ control line on the computer's buss. By loading the counters with different values, via the DIP switch package, and observing the waveform on the scope, these two frequency variables (calculated theoretical and calculated measured) may be correlated. When the counters are inhibited by turning on the inhibit switch of the DIP switch package, no interrupt requests should be issued--the $\overline{IRQ}$ line should stay high.

Testing Results

AVI

Running the prescribed test program, I would expect that the LEDs being individually either on or off should collectively express the binary representation of each character's ASCII code value. In fact this was the result. For example, entering an "at" sign (@) caused this character's ASCII code value (01000000_2) to be represented by the LEDs--all were off except for LED7 (see the parts layout diagram).

When adjusting the micro-potentiometer, as prescribed in the AVI Testing Rationale section, the beep should be so short as to not be discernable with it turned

all the way in one direction. As it is turned in the other direction, the beep should first appear as a chirp then should increase in duration until, at the other end of its adjustment, the beep lasts for a little more than a second. This function was verified by experiment--each entered character caused the oscillator to beep for a variable duration according to the position of the micro-potentiometer.

PIFG

The two frequency variables described in the PIFG Testing Rationale section should, ideally, be equal to one another. By switching the most significant bit switches of the DIP switch package, the interrupt frequency required for a particular application (for instance, 20 Hz) can be approximated. Then, by manipulating the lesser significant bits, the interrupt frequency may be "fine-tuned." By observing the $\overline{IRQ}$ waveform on the scope (see the PIFG Testing Rationale section), the interrupt frequency may be calculated.

The Computer Science Department's resident engineer, John Giles, and I measured the baud rates on the system's buss with an oscilloscope and found them to be slightly off. This may have been due to an improper crystal being used in the system's timing signal generation

circuit (I seem to remember John later replacing these crystals in several of the SWTPC systems with the proper frequency prescribed by SWTPC). Since the baud rates on the buss were found not to be exactly what they say they are (110, 150, 300, 600, 1200), I found that trusting a calculated divisor without fine-tuning the frequency is not advisable. The frequency should always be verified and, if necessary, calibrated. Lastly, when the inhibit switch on the DIP switch package is on, the interrupt frequency does indeed stop--the $\overline{IRQ}$ line stays high.

Comments

PIFG

Concerning measuring the interrupt frequency with an oscilloscope: the accuracy and, with my poor technique, inherent parallax of the scope limit my "fine-tuning" ability. An accurate digital frequency counter would be preferrable with this type of calibration.

AVI

An interesting thing occured when I initialized the B data direction register with all zeros instead of all ones (see the initialization program in the PIFG

Functional Description section). Apparently, this configured the B data register as an input register but I was using it as an output register. This had the effect of inverting the data at the outputs. In the example sited in the AVI Testing Result section, entering an @ sign would cause all of the LEDs to be on except for LED7 instead of the inverse--right--way.

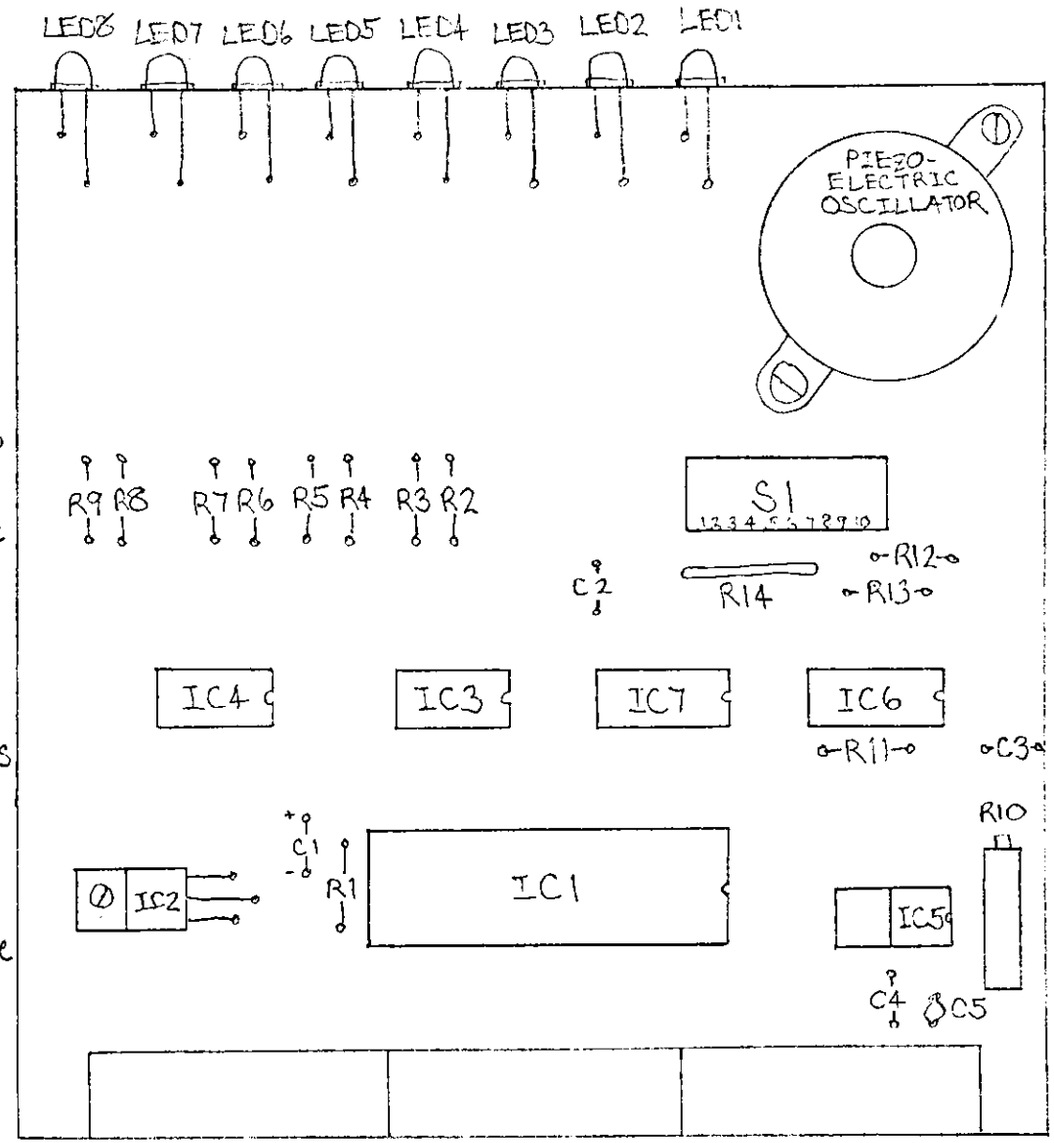
Another, even stranger, phenomenon occurs only during extremely humid days. After pressing the reset button on the computer, the piezo-electric oscillator will chirp like a cricket continuously, without being explicitly activated, until the PIA is initialized.

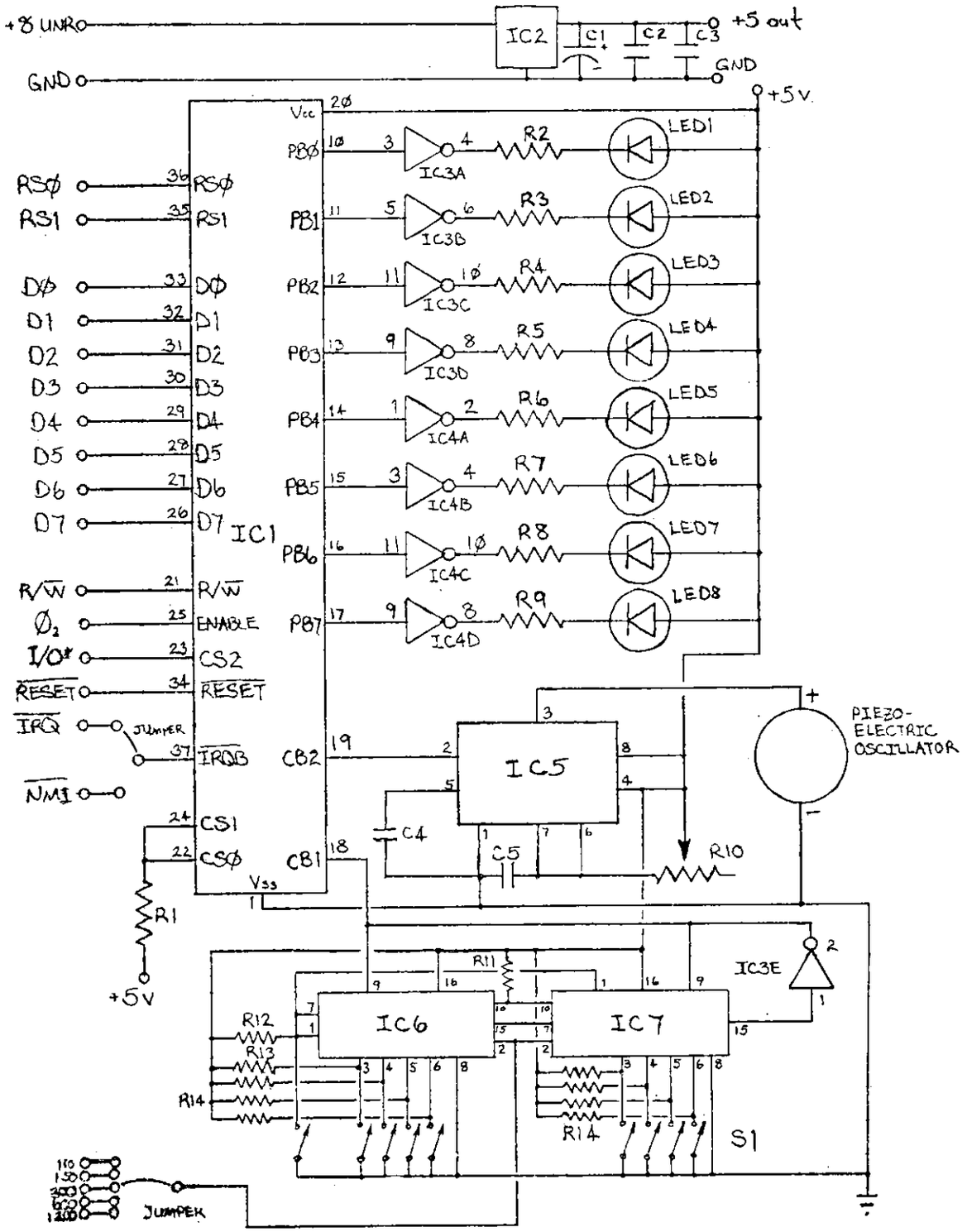
PIFG / AVI
LAYOUT AND PARTS LIST

Parts List

R1, R11, R12, R13 1K ¼W resistors
R2-R9 220Ω ¼W resistors
R10 1M trim potentiometer
R14 1K 7-resister pack
C1 6.8μF electrolytic capacitor
C2-C4 .1μF capacitor
C5 1μF capacitor
LED1-LED8 red light-emitting diodes
Piezo-electric oscillator
S1 10 SPDT switches in DIP package
IC1 6820 PIA
IC2 7805 voltage regulator
IC3, IC4 7404 hex inverters
IC5 555 timer
IC6, IC7 74161 4-bit binary counters

Misc: 3 Female Molex 10-pin sockets, 1 40-pin DIP socket, 3 14-pin DIP sockets, 2 16-pin DIP sockets, pc board, hardware





PIFG / AVI
SCHEMATIC